SONY

MSX 2+

MSX-BASIC Version 3.0 文法書

HIT BIT

はじめに

この本はMSX-BASIC Version3.0（MSX$_{2+}$-BASIC）の全コマンド，関数につきひとつひとつの使いかたを説明するものです。
初めてコンピューターをお使いになるかたや，まだちょっとコンピューターになじんでいないというかたは，別冊のMSX-BASIC解説書をお読みください。この文法書はすでにBASICについて一通りの知識を持っているかたが，MSX$_{2+}$-BASICを学び，実際にプログラミングする際にお役に立つでしょう。

MSX-BASIC Version3.0を本文中ではMSX$_{2+}$-BASICと表します。

MSXはアスキーの商標です。

目次

第1章　MSX$_{2+}$-BASICの概要 …… 3

定数と変数 …… 4
　定　数 …… 4
　変　数 …… 5
式と演算 …… 7
　式の構成要素 …… 7
　算術式の評価 …… 7
　関係式 …… 7
　論理式 …… 8
　文字式 …… 8
画面の構成 …… 9
　スクリーンモード …… 11
　カラーパレット …… 12
　YJK方式 …… 13
ファイルデバイス …… 15
日本語の入力 …… 16
キーの働きと使いかた …… 19
　エディットキー …… 19
　CTRLキーの機能 …… 20
　ファンクションキー …… 21
　その他の制御キー …… 21
Disk BASICについて …… 22
FM音源について …… 22

第2章　コマンド・関数，エラーメッセージ …… 23

コマンドと関数 …… 24
特別な変数と関数 …… 209
フロッピーディスクを使うための拡張コマンド …… 214
メモリーディスクを使うための拡張コマンド …… 217
FM音源に関する拡張コマンド …… 220
日本語入力を行うための拡張コマンド …… 236
エラーメッセージ …… 250

付録 …… 257

文字 …… 258
　MSX$_{2+}$-BASICで扱える文字 …… 258
　ローマ字変換 …… 270
　キーコード表 …… 271
メモリーマップ …… 272
I/Oポートの割り付け …… 275
MSX-DOSについて …… 277

第1章
MSX$_{2+}$－BASICの概要

定数と変数

MSX$_{2+}$-BASICで扱える定数，変数は次のように分類されます。

定数…常に一定の値を表すもの

MSX$_{2+}$-BASICで扱える定数

- 数値型定数
  - 整数型
    - 10進数表現
    - 16進数表現
    - 8 進数表現
    - 2 進数表現
  - 固定小数点型
  - 浮動小数点型
- 文字型定数

数値型定数

整　　数　　型	−32768から32767の範囲 10進表現：そのまま（例…123, −4567） 16進表現：&Hをつける（例…&HA0） 8進表現：&Oまたは&をつける （例　&O146，&765） 2進表現：&Bをつける（例　&B11010110）
浮 動 小 数 点 型	有効数字6けた（単精度）または14けた（倍精度） 指数部−64以上＋62以下（指数部の基数は10） （例　1.345E−10）

単精度と倍精度

固定小数点型
浮動小数点型
}単精度または倍精度いずれか（初期設定は倍精度）

単　　精　　度 （有効数字6けた）	累乗記号にEを使った場合 例…12.34E3 ！を後ろにつけた場合 例…18.32！
倍　　精　　度 （有効数字14けた）	累乗記号にDを使った場合 例…23.456789D10 何も後ろについていない場合 例…3456789012.34 #を後ろにつけた場合 例…3456789012.34#

文字型定数

“ ”(ダブルクォーテーションマーク)に囲まれた，255字までの文字の並び。文字もスペースも全く含まない文字型定数を特にヌルストリングと呼ぶ。

例
```
"HOW ARE YOU?"
"@980"
""
```
(ヌルストリング)

変数…プログラム中で任意に値を書き換えることができるもの

変数名

- 初めの2文字のみ有効（1文字目は必ずアルファベット）
- MSX_{2+}-BASICの予約語（コマンド名，関数名など）自体あるいは予約語を含む文字列は変数名として使えない。

型宣言

型宣言文字による宣言	DEF文による宣言	宣言される型
%をつける 例：A%	DEFINT 例：DEFINT A	整数型
！をつける 例：B!	DEFSNG 例：DEFSNG B	単精度型
#をつける 例：C#	DEFDBL 例：DEFDBL C	倍精度型
$をつける 例：D$	DEFSTR 例：DEFSTR D	文字型

- 型宣言文（DEFINTなど）が実行された後でも，変数名に異なる型の型宣言文字をつければ，型宣言文字が優先される。

型変換

例	説　明
A%=100.5 PRINT A% 100	整数型に宣言された変数に，宣言された型以外の定数を代入すると，変数の型のとおりに代入される。
B=8!/7! PRINT B 1.1428571428571	演算結果は倍精度で代入される。
B!=8/7 PRINT B! 1.14286	変数に単精度の型宣言文字がついているので，演算結果は単精度で代入される。
PRINT 6.7 OR 4.3 6	論理演算は，データを整数に変換した後に行われる。
A!=1.2345678 B=A! PRINT B 1.23457	すでに単精度型のけた数しか持たないデータを倍精度型の変数に代入しても，けた数はそのままである。

式と演算

MSX$_{2+}$-BASICでは，算術式，関係式および論理式を扱うことができます。

式の構成要素

定数，変数，関数および演算記号。

算術式の評価

演算記号	意　　　味	例	優先順位
＋	加算(X＋Y)	X＋Y	6
−	減算(X−Y)	X−Y	
＊	乗算(X×Y)	X＊Y	3
／	除算(X÷Y)	X/Y	
^	累乗(X^n)	X^2	1
−	符号を変える(−X)	−X	2
¥	整数除算	6.7¥2.3	4
MOD	整数除算の剰余	X MOD 10	5

関係式

2つのデータの大小を比較，結果は真(−1)または偽で与えられる。

演算記号	意　　　味	例
＝	等しい	X＝Y，X$＝Y$
＜	小さい	X＜Y，X$＜Y$
＞	大きい	X＞Y，X$＞Y$
＜＞，＞＜	等しくない	X＜＞Y，X$＞＜Y$
＜＝，＝＜	小さいかまたは等しい	X＜＝Y，X$＝＜Y$
＞＝，＝＞	大きいかまたは等しい	X＞＝Y，X$＝＞Y$

論理式

数値型の定数，変数，関数との間で論理演算を行う。

論理演算…データを整数型に変換し，それを16ビットの2進数とみなして対応するビットごとに演算を行う。

演算記号	ビットごとの論理演算の結果
NOT（否定）	<table><tr><td>X</td><td>NOT X</td></tr><tr><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td></tr></table>
AND（論理積）	<table><tr><td>X</td><td>Y</td><td>X AND Y</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>
OR（論理和）	<table><tr><td>X</td><td>Y</td><td>X OR Y</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>
XOR（排他的論理和）	<table><tr><td>X</td><td>Y</td><td>X XOR Y</td></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>
EQV（排他的論理和の否定）	<table><tr><td>X</td><td>Y</td><td>X EQV Y</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>
IMP（含意）	<table><tr><td>X</td><td>Y</td><td>X IMP Y</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>

文字式

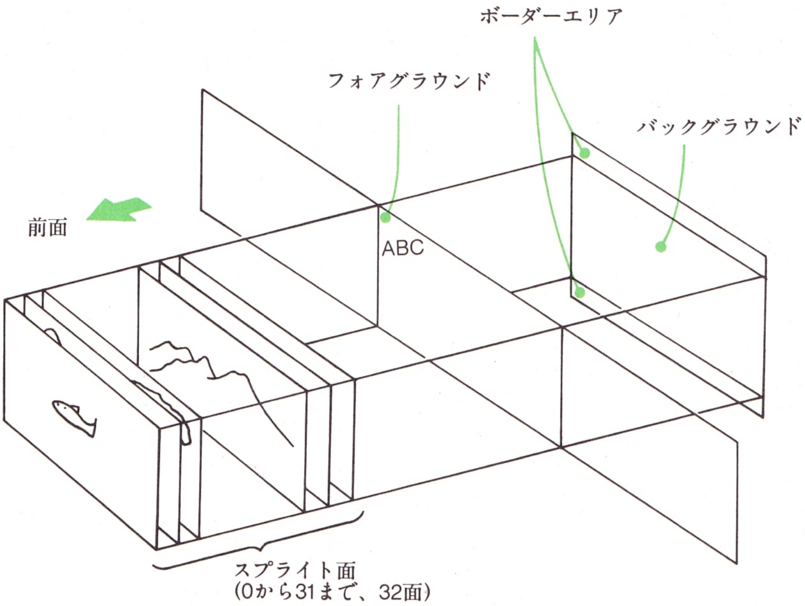
文字型の定数，変数，関数どうしまたはそれらの間の演算，加算のみ有効。

例

```
PRINT "ABC"+"XYZ"
ABCXYZ
```

画面の構成

MSXコンピューターでは表示画面は下図のような構成になっています。実際文字や図形を表示するのがフォアグラウンド面です。フォアグラウンド面の後には，色を変えることはできますが，文字や図形を表示することのできないバックグラウンド面があり，その上下にはボーダーエリアがあります。ボーダーエリアも色を変えることができるだけで，何も表示することはできません。フォアグラウンド面の前には，自由にパターンを定義し，それを動かすことのできるスプライト面が32面あり，32種類のスプライトパターンを使うことができます。



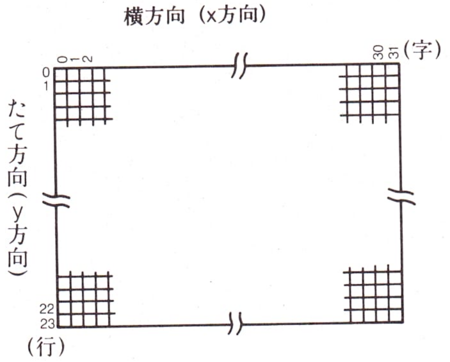
MSX$_{2+}$-BASICでは，ビデオRAMが128Kバイトと豊富にあるため，スクリーンモードによっては，フォアグラウンドに表示するためのデータを数画面分1度に記憶しておくことができます。これを利用し，ビデオRAMを複数のページに分割し，その1ページ分だけを画面に表示するあいだ別のページにデータを書き込むことができるようになっています。BASIC命令が作用してデータが書き込まれるページをアクティブページ，モニターテレビの画面に実際に表示されるページをディスプレイページと言います。
ページの数はスクリーンモードによって異なり，次のようになっています。

スクリーンモード	ページ番号
5	0〜3
6	0〜3
7	0〜1
8	0〜1
10	0〜1
11	0〜1
12	0〜1

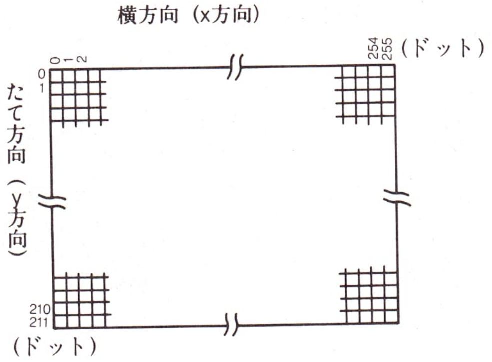
アクティブページとディスプレイページの指定には，SET PAGE文を使います。

画面の座標

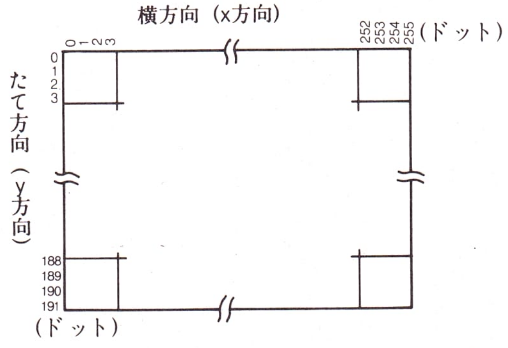
・テキスト画面（スクリーン1の場合）



・グラフィック画面（スクリーン5，10，11，12の場合）



・グラフィック画面（スクリーン3，マルチカラーの場合）



スクリーンモードにより横方向（x方向），たて方向（y方向）の値がかわります。次ページのスクリーンモードの表を参照してください。

SCREENモード

MSX$_{2+}$-BASICでは画面の解像度や表示機能により，次のような12のスクリーンモードがあります。

モード番号	モード	画面表示	カラー	ページ	スプライト
0	テキスト	横最大80字，たて24行 かな表示不可	カラーパレット機能 16色/512色	—	使えない
1	テキスト	横最大32字，たて24行 かな表示可	カラーパレット機能 16色/512色	—	使える
2	グラフィック	256×192ドット	カラーパレット機能 16色/512色 (2色/8ドット)	—	使える
3	グラフィック	64×48ドット マルチカラー	カラーパレット機能 16色/512色	—	使える
4	グラフィック	256×192ドット	カラーパレット機能 16色/512色 (2色/8ドット)	—	使える (強化スプライト)
5	グラフィック	256×212ドット	カラーパレット機能 16色/512色	4ページ	使える (強化スプライト)
6	グラフィック	512×212ドット	カラーパレット機能 4色/512色	4ページ	使える (強化スプライト)
7	グラフィック	512×212ドット	カラーパレット機能 16色/512色	2ページ	使える (強化スプライト)
8	グラフィック	256×212ドット	256色	2ページ	使える (強化スプライト)
10	グラフィック	256×212ドット	表示可能色 12499色 指定可能色 カラーパレット機能 16/512色	2ページ	使える (強化スプライト)
11	グラフィック	256×212ドット	表示可能色 12499色 指定可能色 コード数 256	2ページ	使える (強化スプライト)
12	グラフィック	256×212ドット	表示可能色 19268色 指定可能色 コード数 256	2ページ	使える (強化スプライト)

SCREEN 10，SCREEN 11およびSCREEN 12モードでのカラー表示については13ページをご覧ください。

カラーパレット

スクリーンモード0～7および10では色の表示にカラーパレット機能が使えます。カラーパレットとは，各パレット番号の色に赤，緑，青の輝度を任意に指定して自分の好みに合った色を作ることができる機能です。赤，緑，青それぞれの輝度は0～7の8段階ありますので合成できる色は合計512色になります。カラーパレットはスクリーンモード0～5と7および10では15番までの16色使用可能で，SCREEN6では3番までの4色を使えます。

MSX_{2+}-BASICの初期状態では次のように設定されています。

パレット番号	色	赤の輝度	青の輝度	緑の輝度
0	透明	0	0	0
1	黒	0	0	0
2	緑	1	1	6
3	明るい緑	3	3	7
4	暗い青	1	7	1
5	明るい青	2	7	3
6	暗い赤	5	1	1
7	シアン	2	7	6
8	赤	7	1	1
9	明るい赤	7	3	3
10	暗い黄	6	1	6
11	明るい黄	6	4	6
12	暗い緑	1	1	4
13	マゼンタ	6	5	2
14	灰	5	5	5
15	白	7	7	7

YJK方式

SCREEN 10～12のモードでは，カラーを表示するのにYJK方式と呼ばれる方式を使っています。

ただし，SCREEN 5モードとSCREEN 10モードではBASICの描画コマンドをロジカルオペレーションなしで使用する限り見かけ上の結果は同じになります。ただし，SCREEN 10モードでは同じグラフィックを描くのに2倍のVRAMを必要とします。

また，SCREEN 8モードとSCREEN 11，12モードでの描画コマンドの使いかたはまったく同じですが，SCREEN 11とSCREEN 12モードでは色バケをおこすことがあり，同じプログラムであっても各モードでの実行結果はすべて異なります。

以下にYJK方式とVRAMのデータの関係を説明します。

YJK方式では，画面を横256ドット，たて212ドットに分割します。そして，各ドットに対して1バイトのデータを使ってその色を表わします。このとき，つねに横に連続した4ドットをひと組として色を表わします。

VRAM中の連続する4バイトを，アドレスの小さい順に0，1，2，3とすると，そのデータの内容は次のように表わされます。

ビット / アドレス	7　6　5　4　3	2　1　0
0	Y0	K_L
1	Y1	K_H
2	Y2	J_L
3	Y3	J_H

K_LはKの下位3ビット，K_Hは上位3ビットで，Jについても同様です。

Y0からY3まではそれぞれ5ビットであり，0から31までの32とおりのデータを表わせます。

KおよびJはそれぞれ6ビットで，符号つきで処理するため-32から31までの64とおりのデータを表わせます。

ここで，画面上に表示される4ドットのうち，1番左のドットのR（赤），G（緑），B（青）の輝度をR0，G0，B0とし，2番目から4番目のドットについても同様にそれぞれのR，G，Bの輝度をR1，G1，B1，R2，G2，B2，R3，G3，B3とすると，VRAM上のデータのY，J，Kの成分から，次の式で各輝度が求められます。

1ドット目のR，G，B

$R0 = Y0 + J$

$G0 = Y0 + K$

$B0 = \frac{5}{4}Y0 - \frac{1}{2}J - \frac{1}{4}K$

2ドット目のR，G，B

$R1=Y1+J$

$G1=Y1+K$

$B1=\frac{5}{4}Y1-\frac{1}{2}J-\frac{1}{4}K$

3ドット目のR，G，B

$R2=Y2+J$

$G2=Y2+K$

$B2=\frac{5}{4}Y2-\frac{1}{2}J-\frac{1}{4}K$

4ドット目のR，G，B

$R3=Y3+J$

$G3=Y3+K$

$B3=\frac{5}{4}Y3-\frac{1}{2}J-\frac{1}{4}K$

このようにして色を表わす方式をアトリビュートなしのYJK方式といいます。
VRAMのデータの各バイトのビット3をアトリビュートビットとし，アトリビュートビットが0のときはそのバイトに対応するドットの色はアトリビュートなしのYJK方式と同様の計算式で求め，アトリビュートビットが1のときはJとKの値に関係なく，各バイトの上位4ビットを各ドットの色のパレット番号とみなす方式を，アトリビュート混在のYJK方式といいます。

アドレス＼ビット	7　6　5　4	3	2　1　0
0	Y0	A0	K_L
1	Y1	A1	K_H
2	Y2	A2	J_L
3	Y3	A3	J_H

SCREEN 10～12の各モードでは，次の方式でドットの色を決めます。

SCREEN 10，11:アトリビュート混在のYJK方式
SCREEN 12　　:アトリビュートなしのYJK方式

ファイルデバイス

BASICのコマンドを使ってファイルをアクセスする場合，どのデバイスを使用するかを，デバイス名を使って指定します。扱えるファイルデバイスには次のものがあります。

CAS：……カセットテープ
CRT：……テキストモードの画面
GRP：……グラフィックモードの画面
LPT：……プリンター
MEM：‥‥メモリーディスク
A：～H：……ディスクドライブ名

メモリーディスクとは

カセットテープやフロッピーディスクはプログラムフィルムやデータファイルを記録(セーブ)しておくための記憶装置として使われます。MSX_{2+}-BASICは，さらに内部メモリーの一部を，ファイルをセーブする記憶装置としてカセットテープやフロッピーディスクと同様の方法で扱うことのできる機能を備えています。記憶装置として扱えるメモリーをメモリーディスクと呼びます。

メモリーディスクの容量は1023～32767バイトの範囲で任意に設定可能で，この容量内でなら32個までのファイルを作成することができます。

ただし，メモリーディスク内のファイルはコンピューターの電源を切るか[RESET]ボタンを押すと消滅しますので注意が必要です。

日本語の入力

MSX$_{2+}$-BASICでは，コンピューターに漢字ROMが内蔵されていれば漢字を含む日本語文字列を扱うことができます。(JIS第二水準の漢字ROMが内蔵されている場合，第二水準の漢字も扱うことができます。)

かな漢字変換

MSX-JEを内蔵または取り付けたコンピューターでは，連文節変換によるかな漢字変換が可能です。MSX-JEが取り付けられていない場合でも，単漢字変換により日本語文字列の入力ができます。単漢字変換の方法についてはお使いになる機器の説明書をお読みください。(MSX-JEのカートリッジとして，ソニーHBI-J1などがあります。)

日本語入力モードへの切り換え

CALL KANJI文を実行すると日本語入力モードになります。このとき，すべての変数およびスタック(FOR-NEXT文およびGOSUB/RETURN文用)はクリアされます。また，CLEAR文によりBASICで使用するメモリーの上限番地が設定されていてもクリアされ，新たに漢字処理用の領域が設定されます。

かな漢字変換モードへの切り換え

CALL KANJI文実行後も，英数字の入力モードが継続します。この状態ではBASICの命令語など，日本語文字列を除くプログラムを入力することができます。日本語文字列を入力するためには，さらにかな漢字変換モードへ切り換える必要があります。かな漢字変換モードへ切り換えるには，[CTRL]キーを押しながらスペースバーを押します。(または[GRAPH]キーを押しながら[SELECT]キーを押します。)

もう一度[CTRL] ＋ スペースバー(または[GRAPH]+[SELECT])を押すとかな漢字変換モードは解除されます。

かな漢字変換モードで入力した全角のアルファベットや数字では，BASICの命令語などプログラムを書くことはできません。

日本語入力モードを解除するには，CALL ANK文を実行します。

かな入力モード

かな漢字変換モードでのかな入力方法には，かな直接入力とローマ字入力の2つの方法があります。

かな直接入力ではキーボード上のかなの表記どおりに入力します。ローマ字入力ではかなをローマ字の読みにしたがって入力します。

また，押したキーのアルファベットが入力される状態をアルファベット入力状態と呼びます。

連文節変換

連文節変換では，まとまった文章をひらがなで入力し，それを文節ごとに漢字かな混じりの文字に変換します。文節とは，それ以上区切ると，それだけでは意味がなくなるものが出てくるような，文の単位です。

入力文字の切り換え

MSX-JEを内蔵した（または取り付けた）コンピューターで連文節変換を行う場合，かな漢字変換モードでの入力文字の切り換えは次のキー操作によって行います。

[かな]キーを押してランプを点けた状態:かな直接入力になる。

[SHIFT]キーを押しながら[かな]キーを押してランプを点けた状態:ローマ字入力になる。

[かな]キーのランプが消えた状態:[F1]キーでアルファベット入力とローマ字入力を切り換える。

かな漢字変換の手順

日本語入力モードで，カーソルを漢字入力位置に合わせ，かな漢字変換モードに切り換えたら次のように操作します。

1 希望のかな入力モード（かな直接入力またはローマ字入力）に切り換える。
2 かなで文章を入力する。
3 スペースバーを押す。(変換)
4 変換の対象となっている部分（背景が明るい青）が希望どおりに変換されていれば[RETURN]キーで確定する。
  希望の漢字に変換されていなければスペースバー，[△]，[▽]，[GRAPH]キーで別の候補を表示させ，希望どおりに変換されたら[RETURN]キーで確定する。
5 次の文節を同様の操作で希望の漢字に変換，確定する。

変換時のキー操作

働き	キー	意味	
		変換前	変換中
変換	スペースバー	変換開始	次の候補に進める。
	[GRAPH]キー	—	前の候補に戻す。
	[△]キー	変換開始	変換対象を1ブロック長くする。
	[▽]キー	変換開始	変換対象を1ブロック短くする。

確定	RETURNキー	ひらがなのままで確定する。	変換中の部分を確定する。
全確定	F5キー	入力中の文字を，現在表示されている状態で全部確定する。	
カタカナ変換	F2キー	入力中の文字を全角のカタカナに変換する。(漢字もカタカナになります。)	
半角変換	F3キー	入力中の文字を半角に変換する。(ひらがなと漢字は半角のカタカナになる。)	
コード変換	F4キー	入力されたJISコードを，文字に変換する。	
カーソル移動／変換中断	◁ ▷キー	カーソルを1字分左右に移動する。	すべてひらがなに戻す。
	SHIFT+◁ SHIFT+▷	カーソルを左端または右端に移動する。	すべてひらがなに戻す。
後退削除	BSキー	カーソルの1字前の文字を削除。	—
削除	DELキー	カーソルの位置の文字を削除。	—
全角スペース	SHIFT ＋ スペースバー	全角のスペースを入力する。	次の候補に進める。
モード切り換え	F1キー	アルファベット入力とローマ字入力を切り換える。	
作業中断／変換中断	ESCキー	すべての文字を消す。	すべてひらがなに戻す。
	CTRL+STOPキー	すべての文字を消す。	

キーの働きと使いかた

エディットキー

MSX$_{2+}$-BASICは，フルスクリーンエディター機能をもっていますので，画面上のどこへでも自由にカーソルを動かし，文字を修正，追加，削除することができます。そのためのキーがエデットキーで，各キーの機能は次のとおりです。

キー	働き
カーソル移動キー ← ↑ ↓ →	カーソルを，上下左右に1つ動かす。
HOME	カーソルを画面の左上端に移動する。
SHIFT + HOME	画面の文字を全部消してカーソルを画面の左上隅へ移動する。
INS (Insert)	インサートモードにする(カーソルが小さくなる)。それ以後キーボードから入力した文字が，カーソルの位置に挿入される。 もう一度INSまたはRETURNを押すか，カーソル移動キーなどを押すと，通常のモードに戻る。
DEL (Delete)	カーソルのある位置の文字を消し，それ以降の文字を1字分前へつめる。
BS (Back space)	カーソルのある位置の1つ前の文字を消し，カーソル以降の文字を1字分前へつめる。
TAB	カーソルを次のタブ位置まで進め，間に文字があればスペースに変える。

CTRLキーの機能

MSX$_{2+}$-BASICでは，CTRLキーともうひとつのキーを同時押しすることによって特殊な働きをさせる機能があります。

押すキー	働き
CTRL+B	カーソルを単語（スペースで区切られた文字の集り）の先頭に移動する。カーソルが単語の先頭にあるときは，ひとつ前の単語の先頭に移動する。
CTRL+C	入力待ちの状態やAUTOコマンドによる行番号の自動発生を解除し，コマンド待ち状態に戻す。
CTRL+E	カーソル位置からプログラムの行の最後までを消去する。
CTRL+F	カーソルを次の単語の先頭に移動する。
CTRL+G	ピッという音を出す。
CTRL+H	BSキーと同じ。
CTRL+I	TABキーと同じ。
CTRL+J	カーソルを1行下へ移動する。
CTRL+K	HOMEと同じ。
CTRL+L	SHIFT+HOMEと同じ。
CTRL+M	RETURNキーと同じ。
CTRL+N	カーソルを行の最後の文字の次の位置へ移動する。
CTRL+R	INSキーと同じ。
CTRL+U	行全部の文字を消去する。
CTRL+X	SELECTキーと同じ。MSX2-BASICでは未定義。
CTRL+¥	カーソルを右に動かす。（→ と同じ。）
CTRL+[	ESCと同じ。MSX2-BASICでは未定義。
CTRL+]	カーソルを左に動かす。（← と同じ。）
CTRL+^	カーソルを上に動かす。（↑ と同じ。）
CTRL+_	カーソルを下に動かす。（↓ と同じ。）

ファンクションキー

[F1]から[F5]までのキーをファンクションキーと呼びます。それぞれのファンクションキーには，ある文字列を定義することができ，ひとつのファンクションキーを押すと，そのキーに定義された文字例のキーを押したのと同じ働きをします。MSX_{2+}-BASICでは，起動したときにすでに次のような文字例が定義されています。

[F1] color␣
[F2] auto␣
[F3] goto␣
[F4] list␣
[F5] run [RETURN]
[SHIFT]+[F1] color 15, 4, 7 [RETURN]
[SHIFT]+[F2] load"
[SHIFT]+[F3] cont [RETURN]
[SHIFT]+[F4] list.[RETURN]
[SHIFT]+[F5] [SHIFT]+[HOME]run[RETURN]

ファンクションキーの内容は，KEYコマンドにより自由に定義し直すことができます。

その他の制御キー

[STOP]キー

BASICプログラムの実行中であれば，実行を一時停止します。停止中にもう一度おすと，実行を再開します。

[CTRL] + [STOP] キー

BASICプログラムの実行中であれば，実行を中断します。AUTO文により行番号を自動発生しているときには，コマンド待ち状態に戻します。

[SELECT]キーと[ESC]キー

この2つのキーは，MSX_{2+}-BASICでは未定義です。(何の働きもしません。)

Disk BASICについて

MSX_{2+}-BASICを内蔵しているコンピューターにフロッピーディスクドライブユニットを接続すると，Disk BASICをお使いになれます。
Disk-BASICではMSX_{2+}-BASICのすべての機能が使え，ディスクをファイルデバイスとして扱うためのコマンドが拡張されています。
この本では，コンピューターに内蔵されているBASICとDisk BASICの両方について解説します。BASICとDisk BASICとの相違については，各項ごとに記載してあります。

FM音源について

コンピューターにMSX専用のFM音源LSI（MSX-MUSIC）が内蔵されて（または取り付けられて）いる場合，FM音源制御の拡張コマンドが使用できます。

マークについて

本文中に使われているマークは次の箇所につけられています。

マーク	Disk BASIC特有の機能の説明や，表記方法が記されている箇所
マーク	メモリーディスク特有の機能の説明や，表記方法が記されている箇所
漢マーク	日本語入力のための拡張コマンドに関する説明や表記方法が記されている箇所
♪マーク	FM音源制御のための拡張コマンドなどに関する説明や表記方法が記されている箇所

第2章
コマンド・関数，エラーメッセージ

コマンドと関数

この章では，MSX_{2+}-BASICの各コマンドと関数をアルファベット順に説明します。見かたは下の凡例を参照してください。

コマンド，関数名

GET TIME（get time）

（ゲット タイム）

時計から時刻を読み，文字変数に代入する。　反SET TIME　関GET DATE

関連コマンド

反対の働きをするコマンド

形　式

形式

GET TIME X$ [, A]

X$　条文字型定数，変数，配列変数，式。　入力の条件

A　省現在時刻。　入力を省略したとき

働きと使いかた

実行例

コマンド，関数の機能についての補足説明やそのコマンド，関数を使った実行例。

関数については，関数名の前に[関数]と記されています

関　数　ABS（absolute）

アブソリュート

数値データの絶対値を返す。

入力の項目の省略

形式のところで，［　］で囲んである入力項目は省略可能です。

[例]

ＳＣＲＥＥＮ　［モード］，［スプライトの大きさ］，［キークリックスイッチ］，［ボーレート］，［プリンタータイプ］，［インターレースモード］

により，モードとスプライトの大きさだけを指定するときは，

```
SCREEN 2,3
```

以下，カンマも含めて省略可能

プリンタータイプだけを指定するときは，

```
SCREEN ,,,,1
```

カンマは省略不可

入力項目の繰り返し

……は同じものの繰り返しを示します。

[例]

DATA定数［, 定数］……

では，DATAの後に，1行に入力可能な範囲で，いくつでも定数をつなげることができます。

関　数　ABS（absolute）

アブソリュート

数値データの絶対値を返す。

形　式

ABS(X)

X　系数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

X≧0のときはXを，X<0のときは−Xを返します。

実行例

```
PRINT ABS(2)
 2
PRINT ABS (3-10)
 7
```

関　数　ASC（ascii）

エーエスシー

文字列データの先頭1文字の文字コードを返す。　反CHR$

形　式

ASC(X$)

X$　系文字型定数，変数，配列変数，式。

返される値：整数型，10進表現。

実行例

```
PRINT ASC("d")
 100 ——— "d" の文字コード
PRINT ASC("data")
 100 ——— "d" の文字コード
```

関数 ATN（arc tangent）

アークタンジェント

数値データの逆正接（アークタンジェント）の値を返す。 反TAN 関SIN, COS

形式

ATN(X)

X 条数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

ATN関数は，三角関数tanの値がXとなるような角度を表す浮動小数点型の数値を返します。単位はラジアンです。

●結果を度（degree）の単位で得たい場合は180/円周率をかけます。

実行例

```
PRINT ATN(1)
 .78539816339745        単位はラジアン

PRINT ATN(1)*180/3.14159
 45.000038009905        単位は度
```

AUTO（オート）

指定した行番号から，指定した増分で自動的に行番号を発生する。

形　式

AUTO［先頭行番号］［, 増分］

先頭行番号　条0≦番号≦65529。整数型定数。
　　　　　　省0　ただし“, 増分”も省略とすると10。

増分　条1≦増分≦65529。整数型定数。
　　　省10

働きと使いかた

プログラム入力時，行番号をキーインする手間を省きたいときに使います。

- 発生した行番号にすでにプログラムが存在している場合は行番号の右に＊が表示されます。変更したいときはカーソルをいったん＊の上に移動してスペースバーまたは[DEL]キーで＊を消した後，新しい文を入力します。変更せずに残したいときは[RETURN]を押します。
- 自動発生を中止するときは[CTRL]を押しながら[STOP]を押すか，[CTRL]を押しながら[C]を押します。

実行例

```
AUTO 100,50
100 PRINT "12345"
150*
```

150* ――― すでに行番号150が存在していることを示す。

BEEP（beep）

ビープ

ピッという音を鳴らす。　関SOUND，PLAY

形　式

BEEP

実行例

ピッという音を10回続けて鳴らすプログラムです。

```
10 FOR I=0 TO 9
20 BEEP
30 NEXT I
```

関数　BIN$（binary dollar）

バイナリーダラー

数値データの2進表示を文字型データとして返す。　関HEX$，OCT$

形　式

BIN$(X)

X　条−32768≦X＜65536。数値型定数，変数，配列変数，式。

負数の場合は，その値に65536を加えた値と同じ。

返される値：文字型。

実行例

```
PRINT BIN$(100)
1100100

PRINT BIN$(-32768)
1000000000000000
```

BLOAD (binary load)

ビーロード

機械語プログラムをロードする。またはロードして実行する。

🖫ファイル化されたビデオRAMの内容をロードする。 反BSAVE

形　式

BLOAD "[デバイス名] [ファイル名 [.タイプ名]]"

{ [, R] / [, S] } [, オフセット]

デバイス名　条CAS：
　🖫A：, B：, C：, D：, E：, F：, G：, H：
　🖫省現在使用中のディスクドライブ。

ファイル名　条6字以内の文字列。
　🖫8字以内の文字列。
　省最初に見つかったファイルをロード。
　（フロッピーディスク使用時は省略不可）

🖫タイプ名　条3字以内の文字列。
　省ヌルストリング。

Rオプション　省ロードのみ。

🖫Sオプション　省メインメモリーのプログラムをロード。

オフセット　条整数。
　省0

働きと使いかた

BSAVE文でセーブされた機械語プログラムをロードします。このとき，BSAVE文で指定された先頭番地と終了番地の間にロードされます。オフセットを指定するとその値が先頭番地，終了番地に加えられます。

Rオプションを指定すると，ロード終了後プログラムが実行されます。そのときの実行開始番地はBSAVE文で指定された番地です。

🖫Sオプションを指定すると，Sオプション付きのBSAVE文でセーブしたビデオRAMの内容をロードします。

SCREEN 5から12（9を除く）のときは，アクティブページへ内容がロードされます。

実行例

```
 BLOAD "CAS:PROG4"
⊟BLOAD "A:PROG.BIN",R
⊟10 SCREEN 2
 20 BLOAD"a:box",S
 30 GOTO 30
```

BSAVEの項の最後のプログラム例でセーブしたビデオRAMの内容を，行番行20でロードしています。その結果，長方形が画面に表示されます。

BSAVE（binary save）

（ビーセーブ）

指定したメモリー範囲の内容を2進数でセーブする。

[円]指定した範囲のビデオRAMの内容を2進数でセーブする。 [反]BLOAD

形　式

BSAVE "[デバイス名][ファイル名[.タイプ名]]",

先頭番地，終了番地，{[,実行開始番地] / [,S]}

デバイス名 [条]CAS：

[円]A：，B：，C：，D：，E：，F：，G：，H：

[円][省]現在使用中のディスクドライブ。

ファイル名 [条]6字以内の文字列。

[円]8字以内の文字列。

[省]ヌルストリング。

（フロッピーディスク使用時は省略不可）

[円]タイプ名 [条]3字以内の文字列。

[省]ヌルストリング。

先頭番地，終了番地 [条]−32768≦番地≦65535。整数型定数。

実行開始番地 [条]−32768≦番地≦65535。整数型定数。

[省]先頭番地と同じとみなされる。

[円]Sオプション [省]メインメモリーの内容をセーブ。

働きと使いかた

先頭番地から終了番地までのメモリー範囲の内容を2進コードでセーブします。機械語プログラムをセーブするときに使います。

実行開始番地を指定すると，Rオプション付きのBLOAD文でこの機械語プログラムをロードしたときに，指定した番号から実行が開始されます。省略すると先頭番地が実行開始番地とみなされます。

[円]Sオプションを指定すると，指定した範囲のビデオRAMの内容をファイルとしてセーブします。

SCREEN 5から12（9を除く）のときは，アクティブページの内容がセーブされます。

ビデオRAMのセーブ範囲

スクリーンモード	セーブ範囲
2	0～&H3C20
3	0～&H2040
4	0～&H3C20
5	0～&H76A0
6	0～&H76A0
7	0～&HFAA0
8	0～&HFAA0

実行例

```
BSAVE "CAS:PROG4",&HE000,&HE800,&HE100
BSAVE "B:PROG.BIN",&HE000,&HE800
10 SCREEN 2
20 COLOR=(15,2,6,7)
30 LINE(30,30)-(130,100),,BF
40 BSAVE "A:BOX",0,&H3FFF,S
```

このプログラムでは，行番号30で長方形を画面上に描いたあと，行番号40のBSAVE文（Sオプション付き）で，その状態のビデオRAMの全内容（0番地～&H3FFF番地）を“BOX”というファイルでセーブしています。

CALL (call)

コール

拡張コマンドを実行する。

形　式

CALL拡張コマンド［(引数，引数…)］

引数　㊟整数型定数，変数，配列変数，式。
　　　　文字型定数，変数，配列変数，式。

働きと使いかた

ROMカートリッジにより拡張コマンドが供給された場合，CALL文によりその拡張コマンドを実行することができます。

●CALLという文字の代わりに_(アンダーライン) を使用することもできます。

関　数　CDBL (convert to double precision)

シーダブル

数値データを倍精度のデータにして返す。　　㊐CINT, CSNG, FIX

形　式

CDBL(X)

X　㊟数値型定数，変数，配列変数，式。

返される値：倍精度の数値型。

働きと使いかた

CDBL関数により，与えられた数値データは内部的に倍精度のデータとして扱われることになります。

関数 CHR$（character dollar）

キャラクターダラー

指定した数値を文字コードとする文字を返す。 反ASC，関STRING$

形　式

CHR$(X)

X　条0≦X＜256。数値型定数，変数，配列変数，式。

返される値：文字型。

実行例

```
PRINT CHR$(100)
d
PRINT CHR$(1);CHR$(65)
月
```

文字コード表（258ページ）を参照してください。

関 数 CINT (convert to integer)

シーイント

数値データを整数型に変換する。 関CDBL, CSNG, FIX

形 式

CINT(X)

X 条−32768≦X<32768。数値型定数，変数，配列変数，式。

返される値：整数型。

働きと使いかた

数値データXが整数値のときはそのまま，浮動小数点型のときは小数点以下を切り捨てて整数値に変換します。INT関数との違いは，INT関数がXの整数化であるのに対し，CINTはXの変数の型を変換します。結果は同じですが，BASICの内部処理が異なります。

実行例

```
PRINT CINT(9/2)
 4
PRINT CINT(12*200*55)
Overflow
```

CIRCLE (circle)

サークル

グラフィックモードでのフォアグラウンド上に円，楕円，円弧の一部または扇型を描く。

形　式

CIRCLE [STEP] (中心座標), 半径, [カラー], [開始角], [終了角], [縦横比]

中心のX座標，中心のY座標　条−32768≦座標<32768。数値型定数，変数，配列変数，式。

半径　条−32768≦半径<32768。数値型定数，変数，配列変数，式。

カラー　条SCREEN2〜7、10のとき

0≦カラー<16。数値型定数，変数，配列変数，式。

SCREEN8,11,12のとき

0≦カラー<256。数値型定数，変数，配列変数，式。

省現在のフォアグラウンドカラー。

開始角　条$-2\pi \leqq$ 角 $\leqq 2\pi$（単位ラジアン）。数値型定数，変数，配列変数，式。

省0

終了角　条$-2\pi \leqq$ 角 $\leqq 2\pi$（単位ラジアン）。数値型定数，変数，配列変数，式。

省2π

縦横比　条数値型の正の定数，変数，配列変数，式。

省1

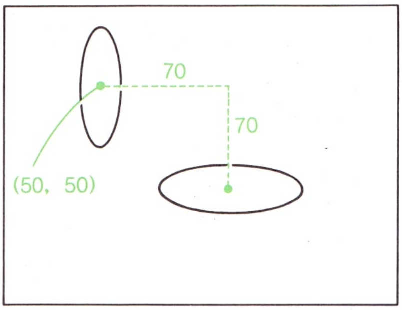
働きと使いかた

指定した座標を中心として，指定した半径で円を描きます。開始角，終了角を指定すると円弧の一部だけを描かせることができます。開始角，終了角に−（マイナス）をつけると扇型を描きます。縦横比で縦方向の半径を横方向の半径の何倍にするかを指定すると楕円を描きます。

STEPを指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標でX，Yが求められます。

実行例

```
10 CLS
20 SCREEN 2
30 CIRCLE (50,50),30,,,,4
40 CIRCLE STEP(70,70),30,,,,.25
50 GOTO 50
```



C

CLEAR（clear）

クリア

数値変数を0に，文字変数をヌルストリングに，配列変数を未定義状態にし，文字列領域の大きさとBASICで使用するメモリーの上限番地を設定する。また，開かれているファイルがあればそれらをすべて閉じる。　関OPEN，CLOSE

形　式

CLEAR［文字列領域の大きさ］［,上限番地］

文字列領域の大きさ　条数値型定数，変数，配列変数，式。
省現在設定されている値（初期状態は200）。ただし文字領域の大きさだけを省略することはできない。

上限番地　条数値型定数，変数，配列変数，式。
条現在設定されている値。

実行例

```
CLEAR 400,55296
```

この文によりすべての変数が初期状態に戻るとともに，文字列用の領域の大きさが400バイトに，BASICプログラムエリアの上限番地が55296番地に設定されます。

CLOAD（cassette load）

シーロード

カセットテープから，BASICのプログラムをロードする。

反CSAVE，関CLOAD?，LOAD

形　式

CLOAD［“ファイル名”］

ファイル名　条6字以内の文字列。7字以上指定すると，7字目以降は無視。
省最初に見つけたプログラムファイルをロード。

実行例

```
CLOAD "PROG1"
```
——PROG1というファイル名のプログラムをカセットテープからメモリーへロード。

● ロード中にエラーが発生した場合は，もう一度テープを巻き戻してロードし直してください。

CLOAD ? (cassette load verify)

シーロードベリファイ

カセットテープにセーブしたプログラムとメモリー上のプログラムを比較照合する。

形　式

CLOAD? [“ファイル名”]

ファイル名　条6字以内の文字列。7字以上指定すると，7字目以降は無視。
　　　　　　省最初に見つかったプログラムファイル。

働きと使いかた

プログラムをセーブしたら，正しくセーブされたかを調べるために実行するコマンドです。このコマンドが実行されると，メモリー上のプログラムと，指定したファイル名でカセットテープにセーブしたプログラムとを比較照合します。

- 比較照合の結果一致すると，OKが表示されて入力待ち状態に戻ります。一致しなかった場合には，Device I/O errorと表示されます。
- ファイル名を省略するか，またはCLOAD?"␣"とすると，テープ上で最初に見つかったプログラムファイルとメモリー上のプログラムとを比較照合します。(␣はブランク(スペース)です。)

実行例

```
CLOAD? "PROG1"
```

CLOSE (close)

(クローズ)

OPEN文で開いたファイルを閉じる。

反OPEN, 関CLEAR

形　式

CLOSE [#] [ファイル番号] [, ファイル番号] …

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数, 変数, 配列変数, 式。

省すべてのファイルを閉じる。

実行例

```
10 MAXFILES=3
20 SCREEN 2
30 OPEN "GRP:" FOR OUTPUT AS #1 ——ファイル1を開く
40 OPEN "GRP:" FOR OUTPUT AS #2 ——ファイル2を開く
50 OPEN "GRP:" FOR OUTPUT AS #3 ——ファイル3を開く
60 PRINT #1,"ABC"
70 PRINT #2,"DEF"
80 PRINT #3,"GHI"
100 CLOSE ——————すべてのファイルを閉じる。
```

CLS (clear screen)

(クリアスクリーン)

スクリーン上のすべての表示を消去する。

形　式

CLS

働きと使いかた

COLOR文で指定されるグラフィックモードのバックグラウンドカラーは, このCLSを実行することによって変化します。

ご注意

日本語入力モードでCLSを実行すると"Illegal function call"のエラーになります。日本語入力モードではCALL CLS文を使用します。

COLOR（color）その1

（カラー）

フォアグラウンド，バックグラウンド，ボーダーエリアの色を指定する。

形　式

COLOR［フォアグラウンドカラー］，［バックグラウンドカラー］，［ボーダーカラー］

フォアグラウンドカラー，バックグラウンドカラー，ボーダーカラー

条SCREEN0～7，10のとき

0≦カラー＜16。数値型定数，変数，配列変数，式。

SCREEN8，11，12のとき

0≦カラー＜256。数値型定数，変数，配列変数，式。

省現在設定されているカラー。

働きと使いかた

SCREEN 8の色番号は次の式で計算することができます。

色番号＝32×（緑の輝度）＋４×（赤の輝度）＋（青の輝度）

0≦緑の輝度≦7。整数。

0≦赤の輝度≦7。整数。

0≦青の輝度≦3。整数。

実行例

COLOR 6 ――フォアグラウンドの色（テキストモードでの文字の色，グラフィックモードでの図形の色）だけが変わる。

COLOR ,2 ――バックグラウンドの色だけが変わる。

COLOR ,,11 ――ボーダーエリアの色だけが変わる。

COLOR 15,4,7 ――初期状態に戻る。

- カラーコード表は12ページをご覧ください。
- 画面の構成については9ページをご覧ください。
- グラフィックモードでは，COLOR文によりバックグラウンドカラーを指定しただけではバックグラウンドの色は変化しません。そのあとでCLSを実行すると初めて色が変わります。

C

COLOR (color) その2

パレット番号に，表示される色を割り当てる。

形　式

COLOR＝（パレット番号，赤の輝度，緑の輝度，青の輝度）

パレット番号　条SCREEN0～7，10のとき

0≦カラー＜16。数値型定数，変数，配列変数，式。

赤，緑，青の輝度　条0≦輝度≦7。整数型定数。

働きと使いかた

赤，緑，青の輝度の組み合わせにより512色まで表すことができます。

このコマンドを実行しないパレット番号の色は初期値（12ページ参照）になります。

実行例

```
100 SCREEN 5
110 FOR I=7 TO 0 STEP -1
120  CIRCLE (120,100),I*10+5,I
130  PAINT (120,100),I,I
140 NEXT I
150 J=(J+1) MOD 8
160 FOR I=0 TO 7
170  COLOR=(I,0,J,0)
180  J=(J+1) MOD 8
190 NEXT I
200 GOTO 150
```

ご注意

日本語入力モードでパレットの色を割り当てるときはCALL PALETTE文を使います。日本語入力モードでCOLOR文を使うと"Illegal function call"のエラーになります。(COLOR＝NEW，COLOR＝RESTORE文についても同様です。)

COLOR（color）その3

電源を入れたときと同じ状態にカラーパレットを戻す。

形　式

COLOR［＝NEW］

NEW 省同じ働き。

パレット番号	色	赤の輝度	青の輝度	緑の輝度
0	透明	0	0	0
1	黒	0	0	0
2	緑	1	1	6
3	明るい緑	3	3	7
4	暗い青	1	7	1
5	明るい青	2	7	3
6	暗い赤	5	1	1
7	シアン	2	7	6
8	赤	7	1	1
9	明るい赤	7	3	3
10	暗い黄	6	1	6
11	明るい黄	6	4	6
12	暗い緑	1	1	4
13	マゼンタ	6	5	2
14	灰	5	5	5
15	白	7	7	7

COLOR=RESTORE（color=restore）

カラー　リストア

ビデオRAM内のカラールックアップテーブルの内容を，VDPのカラーパレットレジスターに代入する。

形　式

COLOR=RESTORE

働きと使いかた

BSAVE文をSオプション付きで実行すると，ビデオRAMの内容をカラーパレットの情報とともにディスクにセーブすることができます。BLOAD文をSオプション付きで実行すると，このデータはビデオRAMにロードすることができます。しかし，カラーパレットの情報はルックアップテーブルに設定されるだけで，実際に表示される色はロード前と変わりません。COLOR RESTOREを実行するとこのルックアップテーブル内の情報をカラーパレットに設定し，BSAVE，Sを実行したときと同じ色で表示することができます。

実行例

```
10 SCREEN 2
20 BLOAD "BOX",S
30 COLOR=RESTORE
40 GOTO 40
```

BSAVEの項の最後のプログラム例でセーブしたビデオRAMの内容を，行番号20でロードし，30でBSAVEしたときと同じ色に修正する。

COLOR SPRITE（color sprite）

カラー　スプライト

指定された面のスプライト全体を指定されたパレット番号の色にする。

形　式

COLOR SPRITE（スプライト面番号）=パレット番号

スプライト面番号　条0≦番号<32。数値型定数，変数，配列変数，式。

パレット番号　条0≦番号<16。数値型定数，変数，配列変数，式。

働きと使いかた

SCREEN 4から8および10から12のとき有効です。

カラー　スプライトダラー
COLOR SPRITE$ (color sprite dollar)

スプライトの走査線ごとに色を指定する。

形　式

COLOR SPRITE$ (スプライト面番号)＝"文字式"

スプライト面番号　条0≦番号<32。数値型定数，変数，配列変数，式。

文字式　条16字以下。

働きと使いかた

SCREEN4から8および10から12のとき有効です。

各文字式が1本の走査線に対応し，文字の各ビットは下記のような意味を持ちます。

- b_7　1の場合に，スプライトを32ドット左にずらす。
- b_6　1の場合に，スプライトの優先順位と衝突を無視し，複数のスプライトが重なった場所ではカラーコードのORを画面に表示する。
- b_5　1の場合に，スプライトの衝突を無視する。
- b_4　未使用
- b_2～b_0　カラーパレットコード

文字式が15文字以下の場合，設定されなかった部分は以前の設定値が残ります。

実行例

```
100 SCREEN 5,1
110 FOR I=0 TO 7
120  A$=A$+CHR$(2^I-1)
130  B$=B$+CHR$(I*2+1)
140 NEXT I
150 SPRITE$(0)=A$
160 COLOR SPRITE$(2)=B$
170 FOR I=0 TO 255
180  PUT SPRITE 2,(I,100),,0
190 NEXT I
200 GOTO 170
```

CONT (continue)

コント

プログラムの実行を再開する。

関STOP

形　式

CONT

働きと使いかた

CTRL＋STOPにより中断したプログラム，あるいはプログラム中のSTOP文で中断したプログラムの実行を再開させます。CONT文を実行すると，中断された次の文から実行を再開しますが，INPUT文の実行中に中断すると，その文の初めから実行が開始されます。

COPY（copy）その1

ディスク上のファイルを同一ディスクまたは他のディスク上にコピーする。

C

形　式

COPY“[ドライブ名1] ファイル名[.タイプ名]”

[TO“[ドライブ名2] ファイル名[.タイプ名]”]

ドライブ名1，2　条A：，B：，C：，D：，E：，F：，G：，H：
　　省現在使用中のディスクドライブ。

ファイル名　条8字以内の文字列。

タイプ名　条3字以内の文字列。
　　省ヌルストリング。

働きと使いかた

ドライブ名1のディスクからドライブ名2のディスクへ，内容をファイル単位でコピーします。

1ファイルだけコピーする場合

●同一ディスクへのコピー

```
COPY "ABC.BAS" TO "XYZ.BAS"
```

このように，ドライブ名1，ドライブ名2ともに省略すると，同一ディスク上に別のファイル名でコピーすることができます。この例では，“ABC.BAS”という名のファイルと同じ内容のファイルが，“XYZ.BAS”というファイル名で同じディスク上に作られます。

```
COPY "ABC.BAS"
```

のように，TO以下を省略すると，同一ファイル名で同一ディスク上にコピーをしようとしたことになり，エラーになります。

●別のディスクへのコピー

ディスクドライブが1台の場合

ディスクドライブが1台でも，これを仮想的にA：B：2台のドライブがあるものとみなして使うことができます。

```
COPY "A:ABC.BAS" TO "B:"
```

を実行すると，

```
Insert diskette for drive B:
and strike a key when ready
```
}——ⓐ

というメッセージがあらわれます。ここで，今ドライブに入っているディスクを抜き取り，新しいディスク(フォーマットされているもの)を入れてなにかひとつキーを押します。すると，

```
Insert diskette for drive A:
and strike a key when ready
```
}——ⓑ

というメッセージがあらわれます。新しいディスクを抜き取り，もとのディスクを入れて，なにかキーを押します。そして，ⓐのメッセージがあらわれたら新しいディスクを，ⓑのメッセージがあらわれたらもとのディスクを入れる，という作業をくり返します。OKがあらわれたら終了で，このとき新しいディスクに“ABC.BAS”というファイルが同じファイル名でコピーされています。
なお，くり返しの回数はコピーしたいファイルの長さによって異なります。

```
COPY "A:ABC.BAS" TO "B:XYZ.BAS"
```

のようにして同じ作業をすると，新しいディスク上にコピーされたファイルには，“XYZ.BAS”と名前が付け変えられます。

C

ディスクドライブが2台以上の場合

ドライブ名1，ドライブ名2で，どのディスクからどのディスクへコピーするのかを指定します。

```
COPY "A:ABC.BAS" TO "B:XYZ.BAS"
```

これにより，Aドライブのディスク中の“ABC.BAS”が，Bドライブのディスク上に“XYZ.BAS”のファイル名でコピーされます。

```
COPY "A:ABC.BAS" TO "B:"
```

これは，Aドライブのディスクの“ABC.BAS”を，Bドライブのディスク上に同一ファイル名(“ABC.BAS”）でコピーするための文です。

```
COPY "A:ABC.BAS" TO "XYZ.BAS"
```

のように，ドライブ名2を省略すると，現在使用中のディスクドライブのディスク上にコピーが行われます。

```
COPY "A:ABC.BAS"
```

のように，TO以下を省略すると，現在使用中のディスクドライブのディスク上に，同一ファイル名でコピーが行われます。したがって，上の例では現在使用中のディスクはAドライブ以外でなければなりません。現在使用中のディスクがAドライブだとエラーになります。

複数のファイルのコピー

?(クエスチョンマーク)でファイル名，タイプ名の1字分ならなんでもあらわすことができます。*(アステリスク)は，すべてのファイル名あるいはすべてのタイプ名の代用として使えます。このことを利用して，1回のCOPYコマンドで，複数のファイルをコピーすることができます。

●ディスクドライブが1台の場合

```
COPY "A:*.BAS" TO "B:"
```

により，あるディスク上にあるファイルのうち“BAS”というタイプ名のついたファイルをすべて別のディスク上にコピーすることができます。このとき，ひとつのファイルのコピーでもそうだったように，メッセージにしたがってもとのディスクと新しいディスクを入れ替える作業を，OKがあらわれるまで続けます。

```
COPY "A:ABC.*" TO "B:"
```

とすれば，あるディスクの上の，“ABC”というファイル名のファイルをすべて別のディスクにコピーできます。

```
COPY "A:PROG??.BAS" TO "B:"
```

とすれば，あるディスク上で，ファイル名がPROGとあと2文字で，タイプ名が“BAS”のもの(PROG01. BAS，PROG02.BAS，PROGAB.BASなど）をすべて別のディスク上にコピーできます。

```
COPY "A:*.*" TO "B:"
```

とすれば，あるディスク上のファイルをすべて別のディスク上にコピーできます。
ディスクドライブが1台の場合は，いずれのときもコピーが終了してOKが出るまでメッセージにしたがってもとのディスクと新しいディスクを入れ替える作業をくり返す必要があります。

●ディスクドライブが2台以上の場合
COPYコマンドの使いかたは，ディスクドライブが1台のときと同じです。ドライブ名1でコピーするファイルの入ったドライブを，ドライブ名2でコピーをとりたい新しいディスクの入ったドライブを指定します。

```
COPY "A:*.*" TO "B:"
```

により，Aドライブのディスクのすべてのファイルが，Bドライブのディスクにコピーされます。

C

COPY（copy）その2

ビデオRAM，配列変数，ディスク上のファイルの間で，画像データを転送します。

形式1

COPY（始点座標）－［STEP］（終点座標）［, ソースページ］TO（転送開始座標），［デスティネーションページ］，［ロジカルオペレーション］

形式2

COPY（始点座標）－［STEP］（終点座標）［, ソースページ］TO
{ "［ドライブ名］ファイル名［. タイプ名］"
  配列変数 }

形式3

COPY { "［ドライブ名］ファイル名［. タイプ名］"
  配列変数 }［, 向き］TO（転送開始座標），［デスティネーションページ］，［ロジカルオペレーション］

形式4

COPY "［ドライブ名］ファイル名［. タイプ名］" TO 配列変数

形式5

COPY 配列変数 TO "［ドライブ名］ファイル名［. タイプ名］"

C

始点座標，終点座標　条−32768≦座標<32768。数値型定数，変数，配列変数，式。

ソースページ，デスティネーションページ　条SCREEN5，6

0≦ページ<4。

数値型定数，変数，配列変数，式。

SCREEN7，8，10，11，12

0≦ページ<2。

数値型定数，変数，配列変数，式。

省アクティブページ。

ロジカルオペレーション　条PSET, PRESET, XOR, OR, AND, TPSET, TPRESET, TXOR, TOR，TAND

省PSET

配列変数　条数値型。

向き　条0≦向き<4。数値型定数，変数，配列変数，式。

省0

転送開始の座標　条−32768≦座標<32768。数値型定数，変数，配列変数，式。

ドライブ名　円条A：，B：，C：，D：，E：，F：，G：，H：

円省現在使用中のディスクドライブ。

C

働きと使いかた

SCREEN5から8および10から12のとき有効です。

ソースページとは転送元のページ，デスティネーションページとは転送先のページのことです。

向きは次のようになります。

0	左上から右下
1	右上から左下
2	左下から右上
3	右下から左上

配列変数はコピーしようとする画像データをすべて記憶するために必要な大きさでなくてはなりません。その大きさは次の式で求められます。

INT（((((ABS（終点X座標－始点X座標)＋1）＊（ABS（終点Y座標－始点Y座標)＋1）＊ピクセルサイズ＋7)／8)＋4)／データサイズ)＋1

●ピクセルサイズ(画面上の1ドットがビデオRAMで何ビットに相当するか)はスクリーンモードにより異なります。

スクリーンモード	ピクセルサイズ
5，7	4
6	2
8，10，11，12	8

●データサイズは用意する配列変数の型により次のようになります。

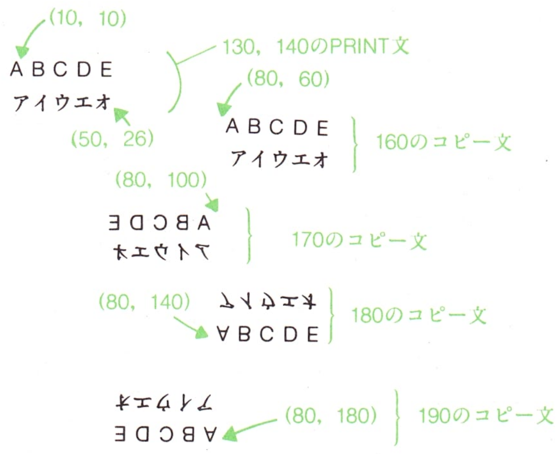
	データサイズ
整　数	2
単精度	4
倍精度	8

通常のデータサイズは8です。

実行例

```
100 SCREEN 5
110 DIM A(89)
120 OPEN "GRP:" FOR OUTPUT AS #1
130 PRESET(10,10):PRINT#1,"ABCDE"
140 PRESET(10,18):PRINT#1,"ｱｲｳｴｵ"
150 COPY (10,10)-(50,26),0 TO A
160 COPY A,0 TO (80,60),0
170 COPY A,1 TO (80,100),0
180 COPY A,2 TO (80,140),0
190 COPY A,3 TO (80,180),0
200 GOTO 200
```

プログラムの実行結果



実行例 110 DIM A (89) の89の値は上記の式で計算した結果です。INT (((((50−10)＋1) ＊ ((26−10)＋1) ＊4＋7)／8)＋4)／4)＋1＝89

コピー　スクリーン

COPY SCREEN (copy screen)

外部映像信号をデジタイズし，VDPに書き込む。

C

形　式

COPY SCREEN [モード]，[マスク]

モード　条0≦モード<2。数値型定数，変数，配列変数，式。

　　　　省0

マスク　条0≦マスク<256。数値型定数，変数，配列変数，式。

　　　　省255

働きと使いかた

SCREEN5から8および10から12のとき有効です。

デジタイズ用の機器を使って，カメラ入力した画面やテレビ画面をコンピューターに書き込むことができます。

モード0のとき，1フィールドの信号をデジタイズしてディスプレイページに書き込み，1のとき，2フィールドの信号をデジタイズしてディスプレイページの1つ前のページと，ディスプレイページに書き込みます。

マスクでコンピューターに書き込むときの色を指定します。上位3ビットに1が立つと赤が書き込まれ，次の3ビットに1が立つと緑が，下位の2ビットに1が立つと青が書き込まれます。たとえばマスクを255とすると赤，緑，青すべてが書き込まれます。

255を2進数で表すと

```
1 1 1  1 1 1  1 1
└┬┘   └┬┘   └┴── 青が書き込まれる。
  │      └──────── 緑が書き込まれる。
  └─────────────── 赤が書き込まれる。
```

実行例

```
100 '--
110 VDP(7)=&HFF
120 SET VIDEO ,,1
130 COPY SCREEN
140 SET VIDEO ,,0
```

C

関 数 COS (cosine)

コサイン

数値データの余弦（コサイン）の値を返す。 関ATN, TAN, SIN

形 式

COS(X)

X 条数値型定数，変数，配列変数，式。(単位ラジアン)

返される値：−1≦値≦1。浮動小数点型定数。

実行例

```
PRINT COS(3.14/3)
 .500459689000814
PRINT COS(60*3.14/180)
 .500459689000814
```

Xを度（degree）の単位で与えるときはCOS（X＊円周率/180）とします。

CSAVE (cassette save)

シーセーブ

BASICのプログラムファイルをカセットテープにセーブする。 反CLOAD 関SAVE

形 式

CSAVE"ファイル名"［,ボーレート］

ファイル名 条6字以内の文字列。7字以上指定すると，7字目以降は無視。

ボーレート 条1≦ボーレート＜3。数値型定数，変数，配列変数，式。

省1

働きと使いかた

ボーレートは，1を指定すると1200ボー，2を指定すると2400ボーになります。

実行例

```
CSAVE "PROG1"
```

——現在メモリー上にあるBASICプログラムを，PROG1というファイル名でカセットテープにセーブ。

関 数 CSNG (convert to single precision)

シーシングル

数値データを単精度のデータにして返す。 関CINT, CDBL, FIX

C

形 式

CSNG(X)

X 条数値型定数，変数，配列変数，式。

返される値：単精度型。

実行例

```
10 PRINT SQR(3)
20 PRINT CSNG(SQR(3))
RUN
 1.7320508075688
 1.73205
```

関 数 CSRLIN (cursor line)

カーソルライン

カーソルの位置のY座標を返す。

形 式

CSRLIN

実行例

```
10 CLS
20 INPUT A$
30 PRINT A$;
40 CL=CSRLIN
50 LOCATE 0,CL+3:PRINT "END"
```

行番号20でA$に入れた文字データの長さにより，行番号30でそれを表示すると1行になったり複数行になったりしますが，表示後のカーソルのY座標（たて方向の位置）を変数CLに入れ，その値に3を加えた値をY座標として“END”を表示させています。したがって，A$に入ったデータの長さにかかわらず，“END”はその3行下に表示されます。

C

関　数 CVI（convert to integer） CVS（convert to single precision） CVD（convert to double precision）

シーブイアイ / シーブイエス / シーブイディー

文字列を数値データに変換して返す。

形　式

CVI(X$)

X$　条2バイトの数値を表す文字型データ。

返される値：整数型。

CVS(X$)

X$　条4バイトの数値を表す文字型データ。

返される値：単精度型。

CVD（X$）

X$　条8バイトの数値を表す文字型データ。

返される値：倍精度型。

働きと使いかた

ディスク上のランダムアクセスファイルでは，数値データもいったん文字型データに変換されて記録されています。この変換は，MKI$，MKS$，MKD$の関数によって行われますが，逆に，ランダムアクセスファイルから読み込んだ文字データを元の数値データに変換するのがCVI，CVS，CVDの各関数です。CVIは2文字（2バイト）の文字列を整数型データに，CVSは4文字（4バイト）の文字列を単精度の浮動小数点型データに，CVDは8文字（8バイト）の文字列を倍精度の浮動小数点型データに変換します。

DATA (data)

データ

READ文で読み取るデータを与える。　　関READ, RESTORE

D

形　式

DATA定数 [,定数]……

定数　条数値型または文字型。

働きと使いかた

- 1つのDATA文中に複数個のデータを並べるときは，カンマ（,）で区切ります。
- DATA文中のデータは，READ文中の変数と一致する型で，対応する順に並んでいれば，READ文に対してどこにあっても，いくつDATA文を使用してもかまいません。
- 文字型データにカンマ(,)やコロン(:)が含まれているときや，前後にスペースを付ける場合は，ダブルクォーテーションマーク(")で囲みます。

実行例

```
10 CLS
20 SCREEN 2
30 READ A,B,C,D
40 LINE (A,B)-(C,D)
50 DATA 0,0,255,191
60 GOTO 60
```

DEF FN (define function)

（デフ エフエヌ）

ユーザー関数を定義する。

D

形　式

DEF FN関数名 [(パラメーター [,パラメーター] ……)]=式

関数名　条数値型，文字型変数。(型は式と一致)

パラメーター　条最大9個までの変数。

式　条数値型，文字型の定数，変数，配列変数，式。

実行例

```
10 DEF FNA(X,Y)=(X*2+Y*3)/(X-Y)
20 B=FNA(4,2)
30 PRINT B
RUN
 7
```

行番号10で関数FNA(X, Y)が続く式のとおりに定義されます。行番号20では，パラメーターX, Yにそれぞれ4, 2の値を与えてこの関数を呼び出しています。結果の7が変数Bに代入されます。

DEFINT (define integer)
DEFSNG (define single precision)
DEFDBL (define double precision)
DEFSTR (define string)

デフイント / デフシングル / デフダブル / デフストリング

変数名の頭文字と変数の型の対応を定義する。

(INT：整数型，SNG：単精度型，DBL：倍精度型，STR：文字型)

形　式

DEFINT文字［一文字］［,文字［一文字］］……

DEFSNG文字［一文字］［,文字［一文字］］……

DEFDBL文字［一文字］［,文字［一文字］］……

DEFSTR文字［一文字］［,文字［一文字］］……

文字　係アルファベットの1文字。

働きと使いかた

先頭の文字が指定した範囲の変数の型を定義します。例えばDEFINT A—Cを実行すると，先頭の文字がA，B，Cのどれかである変数はすべて整数型として扱われます。

●型宣言文字（% ! # $）の優先

DEFINT　A　を宣言しても，後でA#とすればA#は倍精度型の変数になります。

実行例

```
10 DEFINT A-C            ――変数AからCまでを整数型に。
20 A=1.23456789#         ┐
30 ABC=1.23456789#       ┘―行10により変数A，ABCは整数型になる。
40 B#=1.23456789#        ――#をつけて，倍精度型に。
50 C!=1.23456789#        ――！をつけて，単精度型に。
60 PRINT A;ABC;B#;C!
RUN
 1  1  1.23456789  1.23457
```

デフユーザー
DEFUSR (define user)

USR関数により呼び出す機械語サブルーチンの先頭番地を指定する。

形　式

DEFUSR [X]＝開始番地

X　条0≦X≦9。整数型定数。

　省 0

開始番地　条0≦番地＜65536。数値型定数，変数，配列変数，式。

実行例

```
DEFUSR1=&HE000
```

これにより，&HE000番地から始まる機械語サブルーチンがUSR 1として定義されます。

● ひとつのプログラム中で，同じユーザー番号(X)に対して何度も先頭番地を定義し直すことができます。

デリート
DELETE (delete)

プログラム中の指定した行を消去する。

形　式

DELETE [行番号] [—行番号]

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

`DELETE 40`	——行番号40を消去。
`DELETE 20-40`	——行番号20から40までを消去。
`DELETE -50`	——先頭行から行番号50までを消去。
`DELETE .`	——ＬＩＳＴ文などで最後に表示された行，エラーなどによって実行が中断された行が消去される。

● 1行だけを消去したいときは，行番号だけを入力して[RETURN]を押しても消去されます。

DIM (dimension)

配列変数の名前，データの型，大きさ，次元を宣言する。　　　　　　　　　　　　反ERASE

D

形　式

DIM変数名（添字の最大値［，添字の最大値］…）［，変数名（　），……］

変数　条数値型または文字型。

添字の最大値　条0以上の整数型定数，変数，配列変数，式。

働きと使いかた

```
DIM A(15)
```
——A(0)からA(15)までの16個の数値型配列変数の領域をメモリー上に準備。変数の初期値は0。

```
DIM B$(2,3)
```
—次のような12個の領域（文字型）を準備。変数の初期値はヌルストリング。

B$(0,0)	B$(1,0)	B$(2,0)
B$(0,1)	B$(1,1)	B$(2,1)
B$(0,2)	B$(1,2)	B$(2,2)
B$(0,3)	B$(1,3)	B$(2,3)

1つのDIM文で複数個の配列変数を定義

```
DIM A(2),B$(4,2),C(3,3)
```

それぞれの変数はカンマで区切ります。

多次元の配列変数

添字の最大値を2個以上指定すると多次元の配列変数になります。

最大255次元まで定義できます。

```
DIM X(3,4,5)
```
……これは3次元

DIM文の省略

DIM文で宣言をせずに配列変数を使うと，添字の最大値は10とみなされます。

DRAW (draw)

ドロー

サブコマンドの指定どおりに，グラフィックモードの画面上に図形を描く。

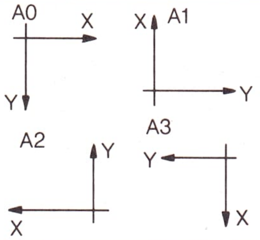
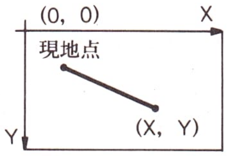
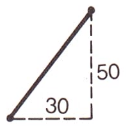
D

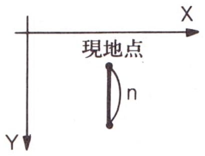
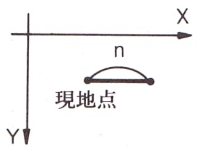
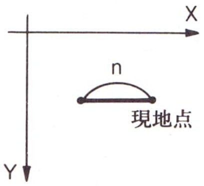
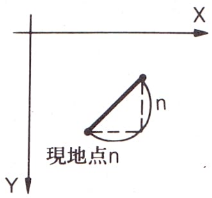
形　式

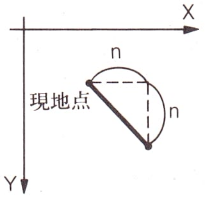
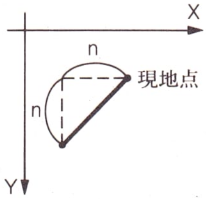
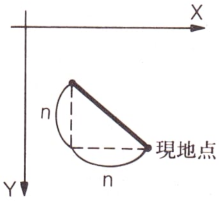
DRAWサブコマンド

サブコマンド　条“　”で囲んだ文字列（定数）または文字列を代入した文字型変数。大文字でも小文字でも良い。

サブコマンド

コマンド	条件	意味	
Sn (scale)	0≦n≦255	線を描くときの1単位のドット数を指定。 n＝1で¼ドット。 初期設定はS4。	
An (angle)	0≦n≦3	基準の座標軸(0°)に対して90°ずつ座標系を回転。 初期設定はA0。	
Cn (color)	0≦n≦15	描く線の色をカラーコードで指定。 初期設定はC15	
Mx, y (move)	0≦x≦255 0≦y≦191	現地点から絶対位置（x, y）まで線を引く。	
M±x,±y (move)	0≦x≦255 0≦y≦191	現地点から横方向へ±x，たて方向へ±y移動。x, yの単位はSコマンドで指定したドット数。	

Un (up)		現地点から,y軸の負の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
Dn (down)		現地点から,y軸の正の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
Rn (right)		現地点から,x軸の正の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
Ln (left)		現地点から,x軸の負の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
En		現地点から, x軸の正, y軸の負方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	

Fn		現地点から，X軸の正，y軸の正の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
Gn		現地点からx軸の負，y軸の正の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	
Hn		現地点から，x軸の負，y軸の負の方向にnだけ離れた地点まで線を引く。 nの単位はSコマンドで指定したドット数。 (省略すると1)	

働きと使いかた

Sn，An，Cnを除くコマンド(線を引くコマンド)では，常に現地点という位置が記憶されています。たとえば，

```
DRAW "M100,120"
```

により，ある位置から（100，120）の位置に向かって直線を引くと，この（100，120）が現地点となります。そして，次に何か線を引くコマンドを与えると，この現地点から指定した点へ向かって線が引かれることになります。

これらの線を引くコマンドには，さらに次の2つのコマンドのいずれかを前につけることができます。

B……現地点は動くが，線が描かれない（例：BM0,0）

N……線は描かれるが，現地点は動かない（例：NU30,30NR30,30）

サブコマンドを変数で表す

```
A$="BM100,150U50E50F50D50L100"
DRAW A$
```

サブコマンドをいったんA＄という文字型変数に代入し，DRAW文ではサブコマンドとして

A＄を指定しています。

サブコマンドの一部を変数で表す（X変数名；）

文字型変数に代入したサブコマンドを，DRAW文の“　”の中で使いたいときは，前にXを，後に；をつけます。

```
A$="U20R20D20L20"
DRAW "BM50,50XA$;"
DRAW "BM150,100XA$;"
```

上の例では，A$に代入したサブコマンドを2つのDRAW文中で使っています。

サブコマンド中のnを変数で表す（＝変数名；）

各サブコマンドで，移動距離や角度，カラーコードなどを表すnは，DRAW文中で定数にしても，変数にしてもかまいません。変数で表すときは，前に＝，後に；をつけます。

```
X=40
DRAW "U=X;"
```

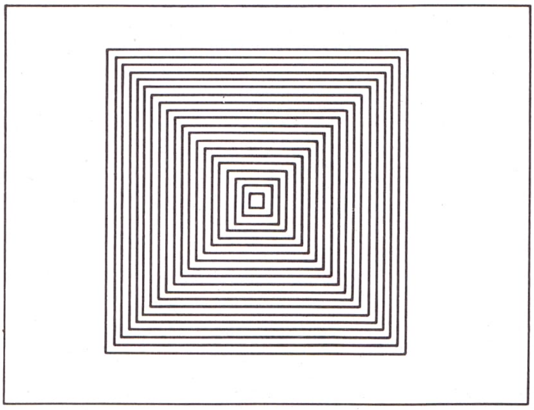
は

```
DRAW "U40"
```

と同じです。

働きと使いかた

```
10 SCREEN 2                      ―― DRAWを使うときはグラフィックモードに
20 DRAW "BM125,100"              ―― 何も描かずに（125，100）へ
30 FOR I=4 TO 240 STEP 12        ┐
40 DRAW "S=I;BURD2L2U2RBD"       ┘ 大きさの異なる正方形を次々と描く
50 NEXT I
60 GOTO 60
```



D

関数 DSKF（disk free）

（ディスクフリー）

ディスクの残り容量をクラスター単位で返す。

形式

DSKF（ドライブ番号）

ドライブ番号　条0≦ドライブ番号<9。数値型定数，変数，配列変数，式。
返される値：整数型。

働きと使いかた

ドライブ番号で指定されたディスクドライブ中のフロッピーディスクの残り容量が，クラスター単位で返されます。ドライブ番号1から順にAドライブ，Bドライブ，Cドライブ……に対応しており，ドライブ番号0は現在使用中のドライブが指定されます。

- 1クラスターは，3.5インチフロッピーディスクの場合2セクターです。また，最大クラスター数は片面ドライブの場合355クラスターです。

END (end)

プログラムの実行を終了させ，コマンド待ち状態に戻す。

このとき，開かれているファイルがあれば，それらをすべて閉じる。

形　式

END

働きと使いかた

END文は，メインプログラムの後にサブルーチンが書かれているときに，メインプログラムの実行終了後再びサブルーチンが実行されるのを防ぐために，メインプログラムの最後に使用します。また，プログラムの実行がいくつかに分岐する場合，それぞれの分岐の終わりにも用います。1つのプログラムの中にいくつあってもかまいません。

● END文で実行が終了したプログラムを再び実行するときはRUNまたはGOTO文を使います。CONT文で再開させることはできません。

```
        .
        .
100 GOSUB 1000
        .
        .
190
200 END
1000 ' SUBROUTINE
        .
        .
1100 RETURN
```

このプログラムの場合，行番号200のEND文がないと，サブルーチンから戻ってきて行番号190が実行された後，GOSUB文なしに行番号1000からのサブルーチンに入ってしまい，その結果行番号1100で“RETURN without GOSUB”のエラーが発生することになります。

E

関数 EOF（end of file）

イーオーエフ

ファイルの最後のデータが読み取られたら−1，それ以外なら0を返す。

形式

EOF（ファイル番号）

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

返される値：整数値（−1または0）。

実行例

```
IF EOF(1) THEN CLOSE #1
```

この文により，ファイル番号1のファイルからデータを読み込み中，最後のデータが読み取られるとファイルを閉じることになります。

ERASE (erase)

イレーズ

配列変数を抹消する。　反DIM

形　式

ERASE　配列変数名　[,配列変数名]……

働きと使いかた

```
10 DIM A(100),B$(4,3)
        .
        .
100 ERASE A,B$
```

この例では，行番号10で宣言した配列変数A，B$を行番号100で抹消しています。抹消後のメモリー領域は，別の目的に使用できるようになります。また，同じ名前の配列変数をDIM文により再定義することもできます。

関　数　ERL (error line)

エラーライン

エラーが発生した行の行番号を返す。　関ERR

形　式

ERL

返される値：数値型。

働きと使いかた

エラーが発生していなければ0，直接コマンドモードでエラーが起こった場合は65535を返します。

ON ERROR文，ERROR文などと組み合わせて使用します。

関数 ERR (error)

エラー

発生したエラーのエラー番号を返す。　関ERL, ERROR

形式

ERR

返される値：整数型。

働きと使いかた

プログラム中で，ERROR文やERL関数と組み合わせてエラー処理などに用いることができます。

●エラーが起こっていない場合は0を返します。

実行例

```
PRINT 10/0
Division by zero
Ok
PRINT ERR
 11
```

ERROR (error)

エラー

指定したエラー番号のエラーを模擬的に発生させる。

またはエラー番号を定義する。 関ERR

形　式

ERROR エラー番号

エラー番号　条0≦番号＜256。数値型定数，変数，配列変数，式。

働きと使いかた

```
ERROR 1
```
——NEXT without FORのエラーを発生させる。（プログラムの実行は停止する）

エラー番号のユーザー定義

```
IF A<0 THEN ERROR 250
```

この文により，変数Aに負の数が代入されるとエラー番号250のエラーが発生します。MSX-BASICでは，すでにエラー番号71までは定義されていますが，それより大きい番号は，ユーザーが自由に定義して使うことができます。

実行例

次のプログラムは，負の数が入力されると正の数を入力するように要求するメッセージを表示し，プログラムの実行を続行するものです。

```
10 ON ERROR GOTO 90
20 FOR I=1 TO 10
30 INPUT A
40 IF A<0 THEN ERROR 250
50 SUM=SUM+A
60 NEXT I
70 PRINT SUM
80 END
90 IF ERR=250 THEN PRINT "Input a positi
ve number.":RESUME 30
100 PRINT "Error!"
```

関　数　EXP（exponential）

エクスポーネンシャル

与えられた数値データXの指数関数e^xを返す。

形　式

EXP(X)

E

X　条X≦145.06286085862。数値型定数，変数，配列変数，式。

返される値：浮動小数点型。

●e（2.7182818284588）は自然対数の底です。

実行例

```
PRINT EXP(100)
 2.6881171418087E+43
```

FIELD (field)

フィールド

ランダムアクセスファイルの入出力レコードの書式を指定する。

形 式

FIELD [#] ファイル番号, 文字長 AS 文字型変数[, 文字長 AS 文字型変数] ……

ファイル番号 条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

文字長 条0≦文字長≦レコード長。数値型定数，変数，配列変数，式。
ひとつのFIELD文中で使われる文字長の合計がレコード長を越えてはならない。

文字変数 条文字型の変数。

働きと使いかた

GET文，PUT文によりランダムアクセスファイルに対してデータの入出力を行う前に，1レコードの中の書式をFIELD文により定義します。

ファイル番号は，すでにOPEN文により開かれているファイル番号を指定します。FIELD文で指定した文字型変数を違う目的の文字型変数として指定すると，FIELD文での指定は無効となり，ファイルへのアクセスはできなくなります。FIELD文で指定する文字型変数と通常の文字型変数は区別してください。

```
FIELD #1,20 AS A$,30 AS B$,40 AS C$
```

このFIELD文により，ファイル番号1の入出力レコードに文字変数A$，B$，C$が割り当てられます。それぞれの長さは20バイト，30バイトおよび40バイトです。このとき，これらの長さの合計（この場合90バイト）が，OPEN文で指定したレコード長を越えるとエラーになります。

なお，指定できるレコード長は，最大256バイトですので，FIELD文で指定する各変数の長さの合計も256バイト以下でなければなりません。

また，ひとつの変数の長さは255バイト以下でなければなりません。

F

F

FILES (files)

ファイルズ

指定したディスク上のファイル名を表示する。

形式

FILES ["[ドライブ名][ファイル名[.タイプ名]]"]

ドライブ名 条A:, B:, C:, D:, E:, F:, G:, H:
省現在使用中のディスクドライブ。

ファイル名 条8字以内の文字列 ┐ ?(1字分), *(ファイル名, タイプ名のいずれか)
タイプ名 条3字以内の文字列 ┘ で置きかえることができる。

実行例

```
FILES "A:TEST.DAT"
```

このFILES文が実行されると，AドライブのTEST.DATというファイルを探し，あればTEST.DATを表示します。なければ“File not found”のエラーメッセージが表示されます。

```
FILES "A:TEST??.DAT"
```

と指定すると，ファイル名の最初の4文字がTESTである6文字までのファイル名でタイプ名に.DATを持つファイル名がすべて表示されます。

```
FILES "A:TEST.*"
```

と指定すると，ファイル名がTESTであるファイル名がすべて表示されます。

```
FILES "A:"
```

では，Aドライブ上のすべてのファイル名が表示されます。

```
FILES "TEST.*"
```

のように，ドライブ名を省略すると，現在使用中のディスクドライブが指定されたものとみなされます。

```
FILES
```

のように，ドライブ名，ファイル名とも省略すると，現在使用中のディスク上のすべてのファイル名が表示されます。

関　数　FIX（fix）

フィックス

数値データの整数部分を返す。　　関CINT，CDBL，CSNG

形　式

FIX(X)

X　条数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

数値データXの小数点以下を切り捨てた値を返します。

実行例

```
PRINT FIX(3);FIX(-3);FIX(3.58);FIX(-3.58)
 3 -3  3 -3
```

FOR-NEXT (for-next)

フォー　ネクスト

FOR文とそれに対応するNEXT文ではさまれたプログラムの実行を繰り返す。

F

形　式

FOR 変数＝初期値 TO 最終値 [STEP増分]

⋮

NEXT[変数]

変数　条数値型。FOR文中とNEXT文中は同じ変数とする。

初期値，最終値　条数値型定数，変数，配列変数，式。

増分　条数値型定数，変数，配列変数，式。

　省1

働きと使いかた

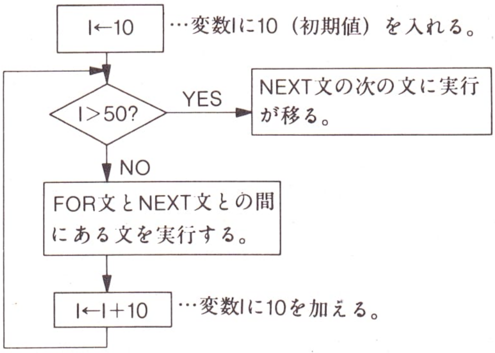
FOR文中で指定した変数の値が，指定した初期値から最終値まで変化する間，FOR文とNEXT文で囲まれたプログラムが繰り返し実行されます。1回実行が終了するごとに変数の値は指定した増分だけ変化します。

● NEXT文中の変数は省略できます。しかし書いておくと，プログラムリストでFORとNEXTの対応がひと目でわかります。

実行例

```
10 FOR I=10 TO 50 STEP 10
20 PRINT "I=";I
30 NEXT I
```

は次のように実行されます。



多重ループ

FOR～NEXTループ内にさらにFOR～NEXTループを持つことができます。この場合は，内側のループは外側のループに完全に含まれていなくてはなりません。それぞれのループには異なる変数を用います。

```
10 FOR I=1 TO 5
20 FOR J=1 TO I
30 PRINT "*";
40 NEXT J
50 PRINT
60 NEXT I
RUN
*
**
***
****
*****
```

内側のループ（20～40）

外側のループ（10～60）

いくつかのFOR文をひとつのNEXT文で終了することができます。この場合，NEXT文中で変数名の省略はできません。変数を内側のループのものから順にカンマで区切って並べます。

```
FOR I=0 TO 10
FOR J=0 TO 5
       .
       .
NEXT J,I
```

関　数　FRE（free）

フリー

MSX$_{2+}$-BASICで使用可能なメモリーの未使用領域のバイト数を返す。

形　式

FRE(X)

FRE("")

X　条任意の数値型定数，変数，配列変数，式。

返される値：整数型。

働きと使いかた

```
PRINT FRE(0)
```
……メモリーの未使用領域のバイト数を表示させる。

```
PRINT FRE("")
```
……メモリーの文字列領域の未使用部分のバイト数を表示させる。

F

GET（get）

ランダムアクセスファイルから1レコードを読み込み，FIELD文によって定義された書式に従って各変数に値を割り当てる。 反PUT

形　式

GET [#] ファイル番号 [,レコード番号]

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

レコード番号　条1≦レコード番号≦65535。整数型定数。
省直前のGET文，PUT文で扱ったレコード番号＋1。

働きと使いかた

```
GET #1,3
```

を実行すると，ファイルから3番目のレコードが読み込まれ，FIELD文によって指定された各変数に値が代入されます。

実行例

```
10 OPEN "A:TEST.DAT" AS #1
20 FIELD #1,2 AS A$,15 AS B$,10 AS C$
30 FOR I=1 TO LOF(1)/255
40 GET #1,I
50 PRINT CVI(A$);" ";B$;" ";C$
60 NEXT I
70 CLOSE #1
80 END
```

GET DATE (get date)

ゲット デート

内蔵の時計から日付けを読み，文字変数に代入する。 反SET DATE 関GET TIME

形 式

GET DATE T$ [, A]

T$ 条文字型定数、変数、配列変数、式。

A 省現在日時。

働きと使いかた

形式はYY／MM／DDでYYは西暦の下2ケタ，MMは月，DDは日を表します。Aを指定するとアラームの日付けを代入します。

実行例

```
10 GET DATE T$
20 PRINT T$
```

GET TIME (get time)

ゲット タイム

時計から時刻を読み，文字変数に代入する。 反SET TIME 関GET DATE

形 式

GET TIME T$ [, A]

T$ 条文字型定数、変数、配列変数、式。

A 省現在時刻。

働きと使いかた

形式はHH：MM：SSでHHは時，MMは分，SSは秒を表します。Aを指定するとアラームの時刻を代入します。

実行例

```
10 GET TIME X$
20 PRINT X$
```

G

ゴーサブ　リターン

GOSUB-RETURN

(go to subroutine—return)

指定したサブルーチンへ実行を移す。RETURN文は，サブルーチンの終わりを示し，GOSUB文の次あるいは指定した行番へ実行を戻す。

関GO TO

形　式

GOSUB行番号

⋮

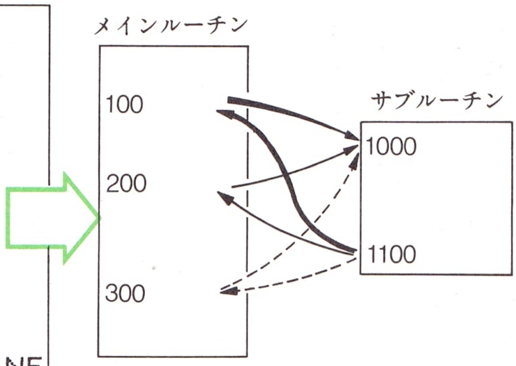
RETURN［行番号］

行番号　条0≦番号≦65529。整数型定数。

省RETURN文中で省略すると，GOSUB文の次の行番号。

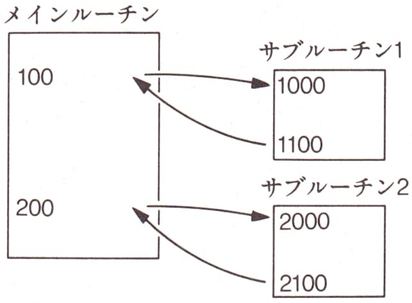
働きと使いかた

```
      .
      .
100  GOSUB 1000
      .
      .
200  GOSUB 1000
      .
      .
300  GOSUB 1000
      .
      .
1000 ' SUBROUTINE
      .
      .
1100 RETURN
```



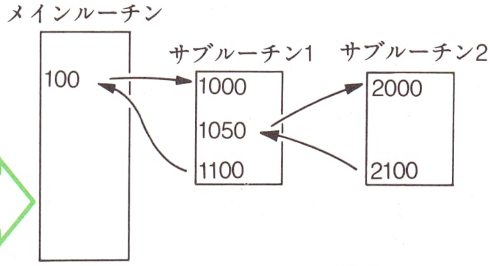
ひとつのサブルーチンを複数のGOSUB文で呼んでいる。

```
         .
         .
100 GOSUB 1000
         .
         .
200 GOSUB 2000
         .
         .
1000 ' SUBROUTINE 1
         .
         .
1100 RETURN
         .
         .
2000 ' SUBROUTINE 2
         .
         .
2100 RETURN
```



ひとつのプログラム中に複数の
サブルーチンを置いている。

```
         .
         .
100 GOSUB 1000
         .
         .
1000 ' SUBROUTINE 1
         .
         .
1050 GOSUB 2000
         .
         .
1100 RETURN
         .
         .
2000 ' SUBROUTINE 2
         .
         .
2100 RETURN
```



あるサブルーチンからさらに他の
サブルーチンを呼んでいる。
(GOSUB文の多重化)

GOSUB文の多重化はメモリーのある限り可能です。

GOTO（go to）

ゴートゥー

指定した行番号へプログラムの実行を移す。

形　式

GOTO行番号

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

GOTO文で指定した行にプログラムの実行が移ります。

●直接コマンドモードで実行すると，指定した行から実行が開始されます。

関　数　HEX$ (hexadecimal dollar)

ヘックスダラー

数値データの16進表示を文字型データとして返す。　関BIN$, OCT$

形　式

HEX$(X)

X　条−32768≦X＜65536。数値型定数，変数，配列変数，式。
負数の場合は，65536にその値を加えた値と同じ。

返される値：文字型。

実行例

```
PRINT HEX$(100)
64

PRINT HEX$(-32768)
8000

PRINT HEX$(255)
FF
```

H

IF-THEN-ELSE (if-then-else)

（イフ ゼン エルス）

式の値によって実行を分岐する。

形　式

IF 式 THEN 文 [ELSE 文]

式　条結果が数値式になる関係式，論理式，算術式。

ELSE文　省式の値が真のときはTHENの後の文へ，偽のときは次の行へ。

働きと使いかた

式の値が真（0以外）ならばTHENの後の文が，式の値が偽（0）ならばELSEの後の文が実行されてから次の行に実行が移ります。

- “ELSE文”を省略すると，式の値が真のときはTHENの後の文が実行されます。偽のときはTHENの後の文は無視され，次の行へ実行が移ります。
- IF-THEN-GOTOという形式ではTHENとGOTOのどちらかを省略できます。

  IF　A=0　THEN　30
  IF　A=0　GOTO　30
  は同じ意味になります。

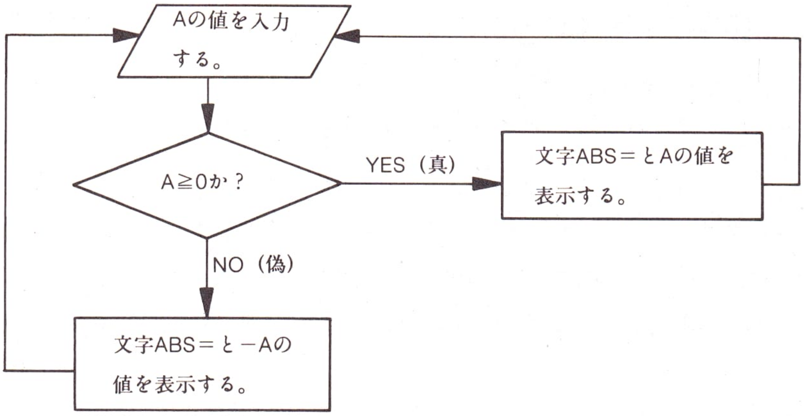
  THENの後は文でも行番号でもかまいません。
  GOTOの後は行番号にします。
- ELSEの後がGOTO文の場合，“GOTO”を省略することができます。
- THENまたはELSEの後に複数個の文を書くと，これらの複数個の文は左から順に実行されます。文と文との間は，コロン（:）で区切ります。

実行例

```
10 INPUT A
20 IF A>=0 THEN PRINT "ABS=";A ELSE PRINT
"ABS=";-A
30 GOTO 10
```

このプログラムは次のように実行されます。

I



IF-THENの多重化

THENやELSEの後に，さらにIF-THEN文を続けることも可能です。
多重化は，1行に書き込める範囲内で何重にもすることができます。

I

関数 INKEY$（inkey dollar）
（インキーダラー）

キーが押されていればそのキーの文字を，押されていなければヌルストリングを返す。

関INPUT

形式

INKEY$

返される値：文字型

働きと使いかた

CTRL + STOP, SHIFT, CTRL以外のキーが押されたら，その文字をデータとして返します。何のキーも押されていなければヌルストリングを返します。

実行例

```
10 CLS
20 PRINT "Press any key."
30 K$=INKEY$
40 IF K$="" THEN GOTO 30    キーが押されるまで繰り返す。
50 PRINT K$;
60 GOTO 30
```

任意のキーが押されるとその文字が変数K$に代入され，行番号50で画面に表示されます。

関数 INP（input）
（インプ）

指定したI/Oポートのデータを読み出す。

関OUT

形式

INP（ポート番号）

ポート番号　条0≦番号＜256。数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

指定したI/Oポートからデータを1バイト入力して返します。

I/Oポートの割り付けについては275ページをご覧ください。

INPUT (input)

インプット

変数の値をキーボードから与える。　関INPUT$, INPUT#, LINE INPUT

形　式

INPUT ["プロンプト文";] 変数 [,変数] [,変数] ……

変数　条数値型，文字型，それらの配列変数。

"プロンプト文"　条文字型定数。

省プロンプト文なしで？のみ表示。

働きと使いかた

キーボードからデータを入力して変数に代入します。このとき，データの前のスペースは無視されます。

- 数値型変数に対するINPUT文の場合，データの途中にあるスペースも無視されます。
- カンマが入力されると，これをデータの区切りとみなし，カンマの前までをひとつのデータとしてひとつの変数に代入します。カンマ自体は代入されません。
- プロンプト文を書くと，データの入力を要求するときにその文が画面に表示されます。プロンプト文を省略すると，？だけが表示されます。
- 変数の数と入力されるデータの数は同じでなければなりません。

実行例

```
10 INPUT A
RUN
?
```

プロンプト文を省略したとき。

```
10 INPUT "A=";A
RUN
A=?
```

プロンプト文を追加したとき。

```
10 INPUT "A AND B ";A,B
RUN
A AND B ? 7
??
```

複数の変数に対し，入力したデータの数が足りないので，？？により足りないデータの入力が要求されている。

```
10 INPUT "A AND B ";A,B
RUN
A AND B ? 1,2,3,4
?Extra ignored
```

変数よりデータを多く入力したときの表示。(余ったデータは無視される。)

関数 INPUT$ (input dollar)

インプットダラー

1.キーボードから，指定した数の文字を入力する。

2.ファイルから，指定した数の文字を入力する。　関INPUT，INPUT#，LINE INPUT

形式

1　INPUT$(N)

2　INPUT$(N,［#］ファイル番号)

N　条1≦N＜256。数値型定数，変数，配列変数，式。

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

返される値：文字型。

I

実行例

```
10 X$=INPUT$(5)
20 PRINT X$
```

行番号10が実行されるとキーボードからの入力待ち状態になり，5文字入力されると変数X$に代入されます。キーボードから入力中は，文字は画面に表示されません。

```
10 OPEN "CAS:TEST" FOR INPUT AS #1
20 X$=INPUT$(50,#1)
30 CLOSE
```

このプログラムでは，カセットテープにセーブされたファイルから50文字入力したところでそれを文字変数X$に代入してファイルを閉じています。

Nの範囲

初期状態ではNが1以上200以下の範囲外ではエラーになります。CLEAR文で文字列領域の大きさを255以上に設定した場合，1以上256未満の値を指定することができます。

INPUT#（input number）

インプットナンバー

OPEN文で開かれたファイルからデータを順に読み取り，変数に代入する。

関INPUT，INPUT$，LINE INPUT#

形　式

INPUT#ファイル番号，変数　[,変数]　……

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

変数　条数値型または文字型，それらの配列変数

働きと使いかた

ファイルからデータを読み取ります。このとき，データが数値型の場合，データの前に書かれたスペース，リターンコード，ラインフィードコードは無視されます。データが文字型の場合，最初に読み取られた文字からスペース，またはカンマ，リターンコード，ラインフィードコードの前までをひとつのデータとして読み取ります。“　”で囲まれている場合は，“　”の中の文だけをデータとして読み取ります。

I

実行例

```
10 OPEN "CAS:DATA" FOR INPUT AS #1 ——読み取り用にファイルを開く
20 IF EOF(1) THEN GOTO 50
30 INPUT #1,A$:PRINT A$ ——データを読み取り，変数A$に代入して画面に表示
40 GOTO 20
50 CLOSE #1
```

関数 INSTR (in string)

インストリング

文字列の中から指定した文字列を検索し，その位置を返す。

形式

INSTR ([N,] X$, Y$)

N　条0≦N＜256。数値型定数，変数，配列変数，式。

　省1

X$, Y$　条文字型定数，変数，配列変数，式。

返される値：整数型

働きと使いかた

文字列X$の中にあるY$が，左から何番目から始まるかを数値データとして返します。Nを指定すると，X$のN番目の文字から検索を開始します。

実行例

```
PRINT INSTR(3,"WHAT IS THIS?","IS")
 6
```

● Nの値がX$の長さより大きいとき，またはX$がヌルストリングのとき，Y$が見つからないときは0が返されます。

ご注意

日本語入力モードで同様の操作を行うには，CALL KINSTR文を使います。

関数 INT (integer)

与えられた数値データ以下の最大の整数値を返す。 関FIX, CINT

形式

INT(X)

X 条数値型定数，変数，配列変数，式。

返される値：数値型。

実行例

```
PRINT INT(3);INT(-3);INT(3.58);INT(-3.58)
 3 -3  3 -4
```

I

INTERVAL ON (interval on)
INTERVAL OFF (interval off)
INTERVAL STOP (interval stop)

内蔵タイマーによる割り込みを有効，無効または保留にする。

形式

INTERVAL ON――割り込み有効

INTERVAL OFF――割り込み無効

INTERVAL STOP――割り込み保留

働きと使いかた

ON INTERVAL GOSUB 文で内蔵タイマーによる割り込みを宣言したあと，実際に割り込みを有効に（INTERVAL ON），あるいは無効（INTERVAL OFF），保留（INTERVAL STOP）にするためのコマンドです。

KEY（key）

ファンクションキーに文字列を定義する。 関KEY LIST, KEY ON, KEY OFF

形　式

KEY ファンクションキー番号，文字列

ファンクションキー番号　条1≦番号<11。数値型定数，変数，配列変数，式。

文字列　条15字以内の文字列。

働きと使いかた

ファンクションキーに文字を定義すると，そのファンクションキーを押すだけで定義した文字列をキーインしたのと同じことになります。

- ファンクションキー番号は，1から5までが[F1]から[F5]に対応し，6から10までが各ファンクションキーを[SHIFT]キーと同時に押したときに対応します。
- RESETボタンを押すか，電源を切ると定義した内容は消え，初期状態に戻ります。
- 文字以外のコード（リターンコード）などもCHR$関数を使って定義できます。

K

実行例

```
KEY 1,"JAPAN"
```
——[F1]に“JAPAN”を定義

```
KEY 2,"CLS"+CHR$(13)
```
——[F2]にCLS[RETURN]を定義

KEY LIST (key list)

キーリスト

ファンクションキーの内容を表示する。

関KEY, KEY ON, KEY OFF

形　式

KEY LIST

働きと使いかた

このコマンドを実行すると，画面上に，各ファンクションキーに定義されている文字列の内容が表示されます。

実行例

```
KEY LIST
color ———————— F1
auto ———————— F2
goto ———————— F3
list ———————— F4
run ———————— F5
color 15,4,7 —— SHIFT+F1
load" ———————— SHIFT+F2
cont ———————— SHIFT+F3
list. ———————— SHIFT+F4
 run ———————— SHIFT+F5
```

初期状態での実行例です。SHIFTキーとF1キーの同時押しに対しては，color 15，4，7が定義されていることがわかります。

K

KEY ON, KEY OFF (key on, key off)

ファンクションキーの内容表示を出す。または消す。　関KEY, KEY LIST

形　式

KEY ONまたはKEY OFF

働きと使いかた

初期状態では画面の一番下の各ファンクションキーに定義されている文字列が表示されています。この表示を消したいときにはKEY OFFを実行します。

- KEY OFFで表示を消すことにより，この行にもPRINT文などで文字を出力することができるようになります。
- 表示を出すときはKEY ONを実行します。

KEY(n) ON (key(n) on)
KEY(n) OFF (key(n) off)
KEY(n) STOP (key(n) stop)

ファンクションキーによる割り込みを有効，無効または保留にする。

形　式

KEY（ファンクションキー番号）ON――――割り込み有効

KEY（ファンクションキー番号）OFF――――割り込み無効

KEY（ファンクションキー番号）STOP――――割り込み保留

ファンクションキー番号　条1≦番号<11。数値型定数，変数，配列変数，式。

働きと使いかた

割り込みに使うファンクションキーをファンクションキー番号で指定します。

```
KEY(1) ON   ………F1キーによる割り込みを有効に
KEY(2) OFF  ……F2キーによる割り込みを無効に
KEY(3) STOP ……F3キーによる割り込みを保留に
```

K

KILL（kill）

ディスク上のファイルを消去する。

形　式

KILL "［ドライブ名］ファイル名［.タイプ名］"

ドライブ名　係A：，B：，C：，D：，E：，F：，G：，H：
　　　　　　省現在使用中のディスクドライブ。
ファイル名　係8文字までの文字列 ｝？(1字分)，＊(ファイル名，タイプ名
タイプ名　　係3文字までの文字列 ｝のいずれか）で置き換えることができる。

働きと使いかた

指定したファイルをディスクから消去します。

実行例

```
KILL "A:TEST.DAT"
```

このKILL文が実行されると，AドライブのTEST.DATというファイルを探し，あればこのファイルを消去します。

```
KILL "A:TEST??.DAT"
```

を実行すると，ファイル名の最初の4文字がTESTである6文字までのファイル名で，タイプ名に .DATを持つファイルがすべて消去されます。

```
KILL "A:TEST.*"
```

を実行すると，ファイル名がTESTであるファイルがすべて消去されます。

```
KILL "A:*.*"
```

を実行すると，Aドライブのディスク上のファイルがすべて消去されます。

いずれの場合も，ドライブ名の指定を省略すると，現在使用中のディスクドライブが指定されたものとみなされます。

関 数 LEFT$ (left dollar)

レフトダラー

文字データの左から任意の数の文字データとして返す。 関RIGHT$, MID$

形 式

LEFT$(X$, N)

X$ 条文字型定数, 変数, 配列変数, 式。

N 条0≦N＜256。数値型定数, 変数, 配列変数, 式。

返される値：文字型

実行例

```
PRINT LEFT$("MSX-BASIC",3)
MSX
Ok
```

```
PRINT LEFT$("MSX-BASIC",3.8)
MSX
Ok
```

Nが整数値でない場合, 小数点以下が切り捨てられる。

```
PRINT LEFT$("MSX-BASIC",0)

Ok
```

Nが0のときはヌルストリングを返す。

L

ご注意

日本語入力モードで同様の操作を行うには, CALL KMID文を使います。

関　数　LEN（length）

レングス

与えられた文字データの文字数（長さ）を数値データとして返す。

形　式

LEN(X$)

X$　条文字型定数，変数，配列変数，式。

返される値：整数型。

実行例

```
PRINT LEN("CHRISTMAS")
 9
PRINT LEN("THE END")
 7
```

文字列がスペースを含む場合，スペースも1文字として数える。

● 文字列がCHR$関数を含む場合も，それをひとつにつき1文字として数えます。

ご注意

日本語入力モードで同様の操作を行うには、CALL KLEN文を使います。

L

LET (let)

データを変数に代入する。

形　式

[LET] 変数＝X

変数　条数値型，文字型の変数，配列変数。

X　条数値型，文字型の定数，変数，配列変数，式。

働きと使いかた

右辺の値を左辺に代入します。

- 文字型定数の場合はダブルクォーテーション（"）で囲みます。
- LETは省略することができます。
- ある型の数値データを，別の型の数値変数に代入すると，数値データは変数が持つ型に変換されます。

実行例

```
LET N=N+1          ――Nの値を1増やす。
A%=45.6:PRINT A%
 45
A$=3+4
Type mismatch      ――文字型変数に数値型データを代入しようとしたのでエラーが発生。
```

L

LINE (line)

グラフィックモードのフォアグラウンド上に直線または四角形を描く。

形　式

LINE [[STEP] (始点座標)] — [STEP] (終点座標),

[カラー] {[, B] / [, BF]} [, ロジカルオペレーション]

始点のX，Y座標　条−32768≦座標＜32768。数値型定数，変数，配列変数，式。
省直前のグラフィック命令で最後に指定された位置。

終点のX，Y座標　条−32768≦座標＜32768。数値型定数，変数，配列変数，式。

カラー　条SCREEN2〜7，10のとき
0≦カラー＜16。数値型定数，変数，配列変数，式。
SCREEN8，11，12のとき
0≦カラー＜256。数値型定数，変数，配列変数，式。
省現在のフォアグラウンドカラー。

B，BF　省直線を描く。

ロジカルオペレーション　条SCREEN5〜8，10〜12のとき
PSET，PRESET，XOR，OR，AND

働きと使いかた

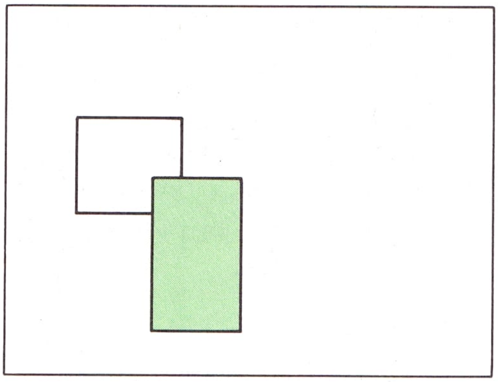
始点座標と終点座標を結ぶ直線を引きます。(B，BFを省略した場合)

- “B”を指定すると，指定の2点を結ぶ直線を対角線とする四角形を描きます。
- “BF”を指定すると，指定の2点を結ぶ直線を対角線とする四角形を描き，囲まれた領域を塗りつぶします。

STEPを指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標系でX，Yが求められます。

実行例

```
10 CLS
20 SCREEN 5
30 LINE (60,60)-(100,100),1,B,AND
40 LINE STEP(-10,-10)-(120,160),8,BF
50 GOTO 50
```



L

LINE INPUT (line input)

（ライン インプット）

254字までの文字列を文字型変数にキーボードから与える。　　関INPUT，LINE INPUT#

形　式

LINE INPUT ［"プロンプト文"；］変数

"プロンプト文"　条文字型定数。

省プロンプト文なしで？のみ表示。

変数　条文字型変数，配列変数。

働きと使いかた

リターンコードだけをデータの区切りとみなして，キーボードから入力された文字列を変数に代入します。INPUT文との違いは，LINE INPUT文では文字列にカンマが含まれていても文字列の一部として代入されることにあります。

実行例

```
10 CLS
20 LINE INPUT "NAME,PHONE? ";N$
30 PRINT N$
RUN
NAME,PHONE? JACK,00-11-22
JACK,00-11-22
```

L

LINE INPUT# (line input number)

（ライン　インプットナンバー）

254字までの文字列をファイルから読み取り，文字型変数に与える。

関LINE INPUT，INPUT#

形　式

LINE INPUT#ファイル番号，変数

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

変数　条文字型変数，配列変数。

働きと使いかた

ファイルから文字型データを読み取りますが，INPUT#文と異なり，スペース，カンマ，ラインフィードコードをデータの区切りとみなさずに，これらも含んだ文字列をひとつの文字列データとして変数に代入します。

データの区切りとみなされるのは，リターンコードだけです。

実行例

```
10 OPEN "CAS:DATA" FOR INPUT AS #1
20 IF EOF(1) THEN GOTO 60
30 LINE INPUT #1,A$
40 PRINT A$
50 GOTO 20
60 CLOSE #1:END
```

すでにDATAというファイル名で次のような手順によりファイルが作成されているとします。

```
PRINT #1,"ABC";",";"DEF"
PRINT #1,"GHI JKL";
PRINT #1,"MNO"
PRINT #1,"PQR"
```

このデータを上のプログラムで読み込んで画面に表示させると，

```
ABC,DEF
GHI JKLMNO
PQR
```

と3つの文字型データとして読み込まれたことがわかります。

L

LIST（list out）

リスト

現在メモリーされているプログラムのリストを表示する。　　関LLIST

形　式

LIST［先頭行番号］［－］［最終行番号］

先頭行番号　条0≦番号≦65529。整数型定数。
　　　　　　条最も小さい行番号。

最終行番号　条0≦番号≦65529。整数型定数。
　　　　　　省最も大きい行番号。

働きと使いかた

画面表示を一時的に停止するためには[STOP]を押します。再開するにはもう一度[STOP]を押してください。中止するためには[CTRL]と[STOP]を押します。

実行例

`LIST`	全部の行を表示。
`LIST 40`	行番号40を表示。
`LIST 20-40`	行番号20から40までを表示。
`LIST -50`	先頭行から行番号50までを表示。
`LIST 30-`	行番号30から最終行までを表示。
`LIST.`	ＬＩＳＴ文などで最後に表示された行，エラーなどによって実行が中断された行が表示される。

L

LLIST (line printer list out)

(エルリスト)

現在メモリーされているプログラムリストを，プリンターで印字する。　関LIST

形　式

LLIST [先頭行番号] [－] [最終行番号]

先頭行番号　条0≦番号≦65529。整数型定数。
　　　　　　省最も小さい行番号。
最終行番号　条0≦番号≦65529。整数型定数。
　　　　　　省最も大きい行番号。

働きと使いかた

指定のしかたはLIST文と同じです。LLIST文実行中は，画面にリストの表示はされません。

● プリンターが接続されていないときや，プリンターが動作可能な状態にないときにLLIST文を実行すると，コンピューターはキーボードからの入力を受け付けない状態のまま停止してしまいます。このようなときはCTRLとSTOPを同時に押すと，入力を受け付ける状態に戻ります。

L

LOAD (load)

ロード

BASICプログラムを指定したデバイスからメモリーにロードする。

[反]SAVE　[関]CLOAD, MERGE

形　式

LOAD"デバイス名 [ファイル名 [.タイプ名]]" [, R]

デバイス名	[条]CAS：
	[FD]A：, B：, C：, D：, E：, F：, G：, H：
	[MD]MEM：
	[省]CAS：
	[FD]現在使用中のディスクドライブ。
ファイル名	[条]6字以内の文字列。
	[MD][FD]8字以内の文字列。
	[省]最初に見つかったファイルをロード。（フロッピーディスク，メモリーディスク使用時は省略不可）
[MD][FD]タイプ名	[条]3字以内の文字列。
	[省]ヌルストリング。
Rオプション	[省]ロードのみ。

働きと使いかた

デバイス名としてCAS：を指定すると，SAVE"CAS：ファイル名"によりカセットテープにアスキー形式でセーブされたプログラムをロードします。

●Rオプションを指定すると，ロード終了後プログラムが実行されます。

実行例

```
 LOAD "CAS:PROG2"
[FD]LOAD "A:PROG.ASC"
```

関数 LOC (location)

エルオーシー

ファイルの中の現在の位置を返す。

形式

LOC（ファイル番号）

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

返される値：整数型。

働きと使いかた

シーケンシャルファイルに対してこの関数を使うと，そのファイルから読み出されたり書き込まれたりしたレコード数が返されます。このとき，1レコードの長さは256バイトです。

ランダムアクセスファイルに対してこの関数を使うと，最後に読み出されたり書き込まれたりしたレコードの番号が返されます。

L

LOCATE（locate）

ロケート

テキストモードのとき，指定の位置へカーソルを移動する。

形　式

LOCATE ［X座標］，［Y座標］，［カーソルスイッチ］

X座標，Y座標　条0≦座標＜256。数値型定数，変数，配列変数，式。

省0

カーソルスイッチ　条0または1

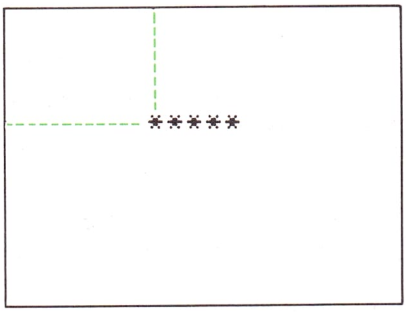
省1

働きと使いかた

カーソルスイッチを1にするとカーソルを表示し，0を指定するとカーソルを表示しません。

実行例

```
10 CLS
20 LOCATE 12,10
30 PRINT "*****"
```



関数 LOF (length of file)

エルオーエフ

指定したファイルの大きさを返す。

形　式

LOF（ファイル番号）

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

返される値：整数型。

働きと使いかた

ファイル番号で指定したファイルの大きさをバイト単位で返します。

関数 LOG (natural logarithm)

ログ

自然対数（log）の値を返す。

形　式

LOG(X)

X　条X＞0。数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

LOG関数は，底をe（2.7182818284588）とする自然対数の値を返します。

●a≠1の正の数aを底とする対数$\log_a b$（b＞0）の値はLOG(b)/LOG(a)で求められます。

実行例

```
PRINT LOG(10)
 2.302585092994
```

関数 LPOS (line printer position)

エルポジション

プリンターバッファ内のプリントヘッドの位置を返す。

形　式

LPOS(X)

X　条任意の数（ダミーの引数)。

返される値：整数型。

働きと使いかた

メモリー内のラインプリンターバッファ中，現在プリンターに出力中の文字の位置を返します。(先頭=0)

L

LPRINT (line print)

エルプリント

式の値をプリンターに出力する。　関PRINT，LLIST

形　式

LPRINT [式] [セパレーター] [式] [セパレーター] [式] ……

式　条数値型，文字型の定数，変数，配列変数，式。

セパレーター　条カンマ(,)　またはセミコロン(;)

省改行する。

働きと使いかた

PRINT文がデータを画面に出力するのに対して，LPRINT文はデータをプリンターに出力します。詳細はPRINTの項を参照してください。

LPRINT USING (line print using)

エルプリント ユージング

指定した書式でデータをプリンターに出力する。　関LPRINT, PRINT USING

形　式

LPRINT USING 書式記号；式[セパレーター][式][セパレーター]……

式　条文字型，数値型の定数，変数，配列変数，式。

働きと使いかた

PRINT USING文が，指定の書式でデータを画面に出力するのに対して，LPRINT USING文は指定の書式でデータをプリンターに出力します。書式記号など詳細はPRINT USINGの項を参照してください。

LSET (left set)

エルセット

ランダムアクセスファイルのレコードにデータを左づめで書き込む。　関RSET

形　式

LSET 文字型変数＝文字式

文字型変数　条文字型の変数。

文字式　条文字型定数，変数，配列変数，式。

働きと使いかた

PUT文でランダムアクセスファイルにデータを書き込む準備として，FIELD文で定義したレコード内の各変数にデータをセットします。LSET文は，データを左づめでセットします。このとき，FIELD文で指定した変数の長さよりもデータが短ければ，余白はスペースで埋められ，データの方が長ければ余った文字データは無視されます。

- 数値データをセットするときは，MKI$，MKS$，MKD$の各関数により，いったん文字型に変換します。

```
LSET A$=X$ ...........文字データX$を，レコード内の変数A$にセット。
LSET B$=MKS$(N) .......数値データNを文字型に変換してレコード内の変数B$にセット。
```

L

MERGE（merge）

マージ

アスキー形式でセーブされたプログラムをロードし，メモリー上のプログラムと合併する。

関LOAD

形　式

MERGE“[デバイス名] [ファイル名　[.タイプ名]]”

デバイス名　条CAS：
　　　　　　　　円A：，B：，C：，D：，E：，F：，G：，H：
　　　　　　　　mMEM：
　　　　　　省CAS：
　　　　　　　　円現在使用中のディスクドライブ。

ファイル名　条6字以内の文字列。
　　　　　　　　m円8字以内の文字列。
　　　　　　省最初に見つかったファイルをマージ。（フロッピーディスク，メモリーディスク使用時は省略不可）

m円タイプ名　条3字以内の文字列。
　　　　　　　　省ヌルストリング。

M

働きと使いかた

SAVE文によりアスキー形式でセーブされたプログラムをロードします。このとき，メモリー上にすでにあるプログラムはそのまま残り，MERGE文でロードされたプログラムと合併します。

- MERGE文によりロードされたプログラムが，すでにメモリーにあるプログラムと同じ行番号を持つ場合には，新たにMERGE文でロードされた方の行が残り，前のプログラムの行は消されます。

実行例

```
 MERGE "CAS:PROG3"
円MERGE "A:PROG.ASC"
mMERGE"MEM:PROG. ASC"
```

関　数　MID$（middle dollar）

ミッドダラー

文字データの一部分をとり出して返す。　　関LEFT$, RIGHT$

形　式

MID$（X$, M [, N]）

X$　　条文字型定数，変数，配列変数，式。

M　　条1≦M＜256。数値型定数，変数，配列変数，式。

N　　条1≦N＜256。数値型定数，変数，配列変数，式。

　　省M字目以降の全文字を返す。

返される値：文字型。

実行例

```
PRINT MID$("JAPANUKFRANCE",6,2)
UK
PRINT MID$("JAPANUKFRANCE",6,2.6)
UK
PRINT MID$("JAPANUK",6,4)
UK
PRINT MID$("JAPANUK",12,5)

PRINT MID$("JAPANUK",6,0)
```

（2番目の例）Nが整数値でない場合，小数点以下が切り捨てられる。

（3番目の例）M字目以降にN字ない場合，M字以降の全文字が返される。

（4・5番目の例）Mの値がX$の長さより大きい場合，あるいはNが0の場合，ヌルストリングが返される。

ご注意

日本語入力モードで同様の操作を行うには，CALL KMID文を使います。

M

MID$=Y$ (middle dollar)

ミッドダラー

文字列の一部を他の文字列で置き換える。 関MID$

形　式

MID$ (X$, M [, N]) = Y$

X$, Y$　　条文字型定数，変数，配列変数，式。

M　　条1≦M<256。数値型定数，変数，配列変数，式。

N　　条1≦N<256。数値型定数，変数，配列変数，式。

　　省X$のM番目以降の文字がY$で置き換えられる。

働きと使いかた

文字列X$の左からM番目以降N文字分を，Y$の先頭からN番目までの文字で置き換えます。ただし，実行後X$の長さは変わりません。

実行例

```
10 X$="ABCDEFG"
20 Y$="QRSTUVWXYZ"
30 MID$(X$,4,2)=Y$
40 PRINT X$
RUN
ABCQRFG
```

M

関 数 MKI$ (make integer dollar) MKS$ (make single precision dollar) MKD$ (make double precision dollar)

エムケイアイダラー MKI$ (make integer dollar)
エムケーエスダラー MKS$ (make single precision dollar)
エムケーディーダラー MKD$ (make double precision dollar)

数値データをその内部表現形式に対応した文字列に変換する。

形 式

MKI$(X)

X 条整数型定数，変数，配列変数，式。

返される値：2文字（2バイト）の文字列。

MKS$(Y)

Y 条単精度型定数，変数，配列変数，式。

返される値：4文字（4バイト）の文字列。

MKD$(Z)

Z 条倍精度型定数，変数，配列変数，式。

返される値：8文字（8バイト）の文字列。

働きと使いかた

ランダムアクセスファイルに書き込むことができるのは文字型のデータだけです。そこで，数値データをランダムアクセスファイルに書き込む前に，MKI$，MKS$，MKD$の各関数により文字型データに変換します。この変換は，LSET/RSET文によりデータをレコード内の指定位置に割り当てるときに行います。

数値データは，その型によりメモリー内を占める大きさが異なります。整数型は2バイト，単精度の浮動小数点型は4バイト，倍精度の浮動小数点型は8バイトとなっています。そこで，たとえば数値データが整数型の場合は

```
LSET A$=MKI$(X)
```
(左づめのとき)

のようにして，整数型数値データXをMKI$関数により，2文字分（2バイト）の文字データに変換してレコード内に割り当てます。したがって，変換後の文字データA$に対しては，あらかじめFIELD文により2バイト以上の長さが確保されていなければなりません。

同様に，MKS$関数は単精度のデータを4文字（4バイト）の文字列データに，MKD$関数は倍精度のデータを8文字（8バイト）の文字列にデータ変換します。ファイルからデータを読み込むときに，文字型に変換されている数値データを元の数値データに戻す関数としてCVI，CVS，CVDの各関数があります。

MOTOR (motor)

モーター

カセットテープレコーダーのモーターをON/OFFする。

形　式

MOTOR {ON / OFF}

働きと使いかた

コンピューターのTAPE端子とカセットテープレコーダーのリモコン端子を接続し，レコーダーを再生または録音状態にしておくと，MOTOR ONでテープが動き，MOTOR OFFで止まります。MOTORだけを実行すると，そのときONならばOFFに，OFFならONに切り換わります。

M

NAME (name)

ディスク上のファイル名を変更する。

形　式

NAME "[ドライブ名] 旧ファイル名 [. 旧タイプ名]" AS "新ファイル名 [. 新タイプ名]"

ドライブ名　条A：，B：，C：，D：，E：，F：，G：，H：
　　省現在使用中のディスクドライブ。
旧ファイル名，新ファイル名　条8字以内の文字列。
旧タイプ名，新タイプ名，　条3字以内の文字列。
　　省ヌルストリング。

働きと使いかた

ファイルの内容を変えずにファイル名を変えるときに使います。

実行例

```
NAME "A:OLD.BAS" AS "NEW.BAS"
```

これにより，Aドライブのディスク上のOLD．BASというファイルの名前がNEW．BASに変わります。

● 旧ファイル名［.旧タイプ名］は，すでにディスク上に存在するものでなければなりません。また，新ファイル名［.新タイプ名］として指定した名前がすでにディスク上に存在していてはいけません。

N

NEW (new)

メモリー上のBASICプログラムを消去し，変数をクリアする。

形　式

NEW

働きと使いかた

新しくプログラムを入力しようとする前に実行してください。前のプログラムがすべて消え，コマンド待ちの状態に戻ります。

● 機械語のプログラムがメモリー上にある場合は，NEW文を実行してもそのまま残ります。

関数 OCT$ (octonary dollar)

オクトダラー

数値データの8進表示を文字型データとして返す。 関BIN$, HEX$

形式

OCT$(X)

X 条−32768≦X<65536。数値型定数，変数，配列変数，式。負数の場合は，65536にその値を加えた値と同じ。

返される値：文字型。

実行例

```
PRINT OCT$(100)
144

PRINT OCT$(65536-32768)
100000
```

O

ON ERROR GOTO (on error go to)

(ルビ: オン エラー ゴートゥー)

エラーが発生したら，指定した行番号へ実行を移す。

形　式

ON ERROR GOTO 行番号

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

プログラム実行中に生じたエラーによって実行が中断するのを防ぐために使います。ON ERROR GOTO文を宣言したあとでエラーが生じると指定した行番号に実行が移ります。(直接コマンドモードでエラーが発生しても指定した行番号に実行が移ります)

実行例

```
10 ON ERROR GOTO 100
20 INPUT A
30 B=SQR(A)
40 PRINT "SQR(A)=";B
50 END ――――――メインルーチンとエラー
                  処理ルーチンを区別するＥＮＤ文

100 IF ERR=5 AND ERL=30 THEN PRINT  ┐
"Input a positive number."          ┘――エラー処理ルーチン
110 RESUME 20
```

ON ERROR GOTO文を無効にするには

ON ERROR GOTO 0を実行します。

O

ON-GOSUB (on-go to subroutine)

(オン ゴーサーブ)

式の値によって，プログラムの実行をいくつかの指定した行番号で始まるサブルーチンに分岐する。

形　式

ON 式 GOSUB 行番号 [, 行番号] ……

式　条0≦式<256。数値型定数，変数，配列変数，式。

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

```
100 ON X GOSUB 500,600,700
```

では，Xの値が1なら行番号500からのサブルーチンに，2なら番号600からのサブルーチンに，3なら行番号700からのサブルーチンに実行が分岐します。サブルーチンからは，RETURN文によりメインプログラムに戻ります。

式の値と実行結果

式の値が整数でない場合……小数点以下が切り捨てられる。

式の値が0，またはGOSUBで指定した行番号の個数より大きい場合……ON～GOSUB文の次の文へ移る。

式の値が負または255より大きい場合…エラーになる

O

ON-GOTO（on-go to）

オン　ゴートゥー

式の値によって，プログラムの実行をいくつかの行番号に分岐する。

形　式

ON 式 GOTO 行番号　[，行番号]　……

式　条0≦式＜256。数値型定数，変数，配列変数，式。

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

```
100 ON X GOTO 120,130,180
```

では，Xの値が1なら行番号120に，Xの値が2なら行番号130に，Xの値が3なら行番号180に分岐します。

式の値と実行結果

式の値が整数でない場合…小数点以下が切り捨てられる。

式の値が0，またはGOTOで指定した行番号の個数より大きい場合…ON～GOTO文の次の文へ移る。

式の値が負または255より大きい場合…エラーになる。

O

ON INTERVAL GOSUB
(on interval go to subroutine)

内蔵タイマーによる割り込みが生じたときに分岐するサブルーチンを宣言する。

形　式

ON INTERVAL＝インターバル時間 GOSUB 行番号

インターバル時間　条−32768≦時間<65536。0を除く数値型定数，変数，配列変数，式。

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

内蔵タイマーにより一定時間ごとに割り込みがかかったときに分岐するサブルーチンの先頭番号を宣言する文です。

割り込みの間隔は，約（インターバル時間×1/60）秒になります。

つまり，インターバル時間を60に指定すると約1秒ごとに割り込みがかかるようになります。

実行例

```
10 ON INTERVAL=60 GOSUB 100
20 INTERVAL ON
30 SCREEN 2,1
40 SPRITE$(1)=CHR$(&H18)+CHR$(&H3C)+CHR$
(&H66)+CHR$(&HDB)+CHR$(&HE7)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
50 GOTO 50
100 X=INT(RND(1)*256):Y=INT(RND(1)*192)
110 C=INT(RND(1)*14)+2
120 PUT SPRITE 1,(X,Y),C,1
130 RETURN 50
```

このプログラムでは，行番号10，20により1秒間隔で割り込みがかかり，行番号100からのサブルーチンへ実行が移ります。サブルーチンではUFOの形のスプライトパターンを表示したあと，RETURN 50により行番号50に戻ります。

● インターバル時間を負数にすると，インターバル時間は65536に指定のインターバル時間を加えたものに等しくなります。

ON KEY GOSUB（オン キー ゴーサブ）
(on key go to subroutine)

ファンクションキーにより割り込みをかけたときに分岐するサブルーチンを宣言する

形　式

ON KEY GOSUB 行番号 [, 行番号] …

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

ファンクションキーにより割り込みをかけたときに分岐するサブルーチンの先頭行番号を宣言する文です。GOSUB以下の行番号はカンマで区切って最大10個まで指定することができ，はじめのものから[F1]，[F2]…のように対応します。

実行例

```
10 ON KEY GOSUB 1000,,2000
20 KEY(1) ON:KEY(3) ON
```

この2行により，[F1]が押されれば行番号1000からのサブルーチンに，[F3]が押されれば行番号2000からのサブルーチンに実行が移ります。
サブルーチンからはRETURN文により戻ります。

O

ON SPRITE GOSUB (オン スプライト ゴーサブ)
(on sprite go to subroutine)

スプライトの衝突により割り込みがかかったときに分岐するサブルーチンを宣言する。

形　式

ON SPRITE GOSUB 行番号

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

スプライトパターン同士の衝突による割り込みがかかったときに分岐するサブルーチンの先頭番号を宣言する文です。

実行例

```
10 ON SPRITE GOSUB 1000
20 SPRITE ON
```

この2行により，スプライトの衝突が起きると行番号1000からのサブルーチンへ実行が移ります。サブルーチンからはRETURN文で戻ります。

O

ON STOP GOSUB
(on stop go to subroutine)

CTRL + STOP キーによる割り込みがかかったときに分岐するサブルーチンを宣言する。

形　式

ON STOP GOSUB 行番号

行番号　省0≦番号≦65529。整数型定数。

働きと使いかた

CTRL + STOP キーによる割り込みがかかったときに分岐するサブルーチンの先頭行番号を宣言する文です。

実行例

```
10 ON STOP GOSUB 1000
20 STOP ON
```

この2行により，CTRL キーと同時に STOP キーが押されれば行番号1000からのサブルーチンに実行が移動します。
サブルーチンからはRETURN文で戻ります。

ご注意

サブルーチンが実行されたら，なんらかの形でプログラムの実行が終了するようにプログラムを作っておく必要があります。次のようなプログラムでは，プログラムを終了させるには RESET ボタンを押すしかありません。

```
10 ON STOP GOSUB 100
20 STOP ON
30 PRINT "MAIN ROUTINE"
40 GOTO 40
100 PRINT "CTRL+STOP EXECUTED"
110 RETURN 30
```

O

ON STRIG GOSUB
(on stick trigger gosub)

コントローラー（スペースバー，ジョイスティック，マウス，トラックボールなど）のボタンによる割り込みがかかったときに分岐するサブルーチンを宣言する。

形　式

ON STRIG GOSUB 行番号［，行番号］…

行番号　条0≦番号≦65529。整数型定数。

働きと使いかた

コントローラーのボタンを押すことにより割り込みがかかったときに分岐するサブルーチンの先頭行番号を宣言する文です。
GOSUB以下の行番号はカンマで区切って最大5つまで指定することができます。

ON STRIG GOSUB行番号❶，行番号❷，行番号❸，行番号❹，行番号❺

行番号❶…スペースバーが押されたときに分岐
行番号❷…スペースバー以外のコントローラーAのボタン1
行番号❸…スペースバー以外のコントローラーBのボタン1
行番号❹…スペースバー以外のコントローラーAのボタン2
行番号❺…スペースバー以外のコントローラーBのボタン2

O

実行例

```
10 ON STRIG GOSUB 1000,2000,3000
20 STRIG(0) ON:STRIG(1) ON:STRIG(2) ON
```

この2行により，スペースバーが押されれば行番号1000からのサブルーチンに，スペースバー以外のコントローラーAのボタン1が押されれば行番号2000からのサブルーチンに，スペースバー以外のコントローラーBのボタン1が押されれば行番号3000からのサブルーチンに実行が移ります。
サブルーチンからはRETURN文により戻ります。

OPEN（open）

ファイルを開く。

【反】CLOSE　【関】MAX FILES

形　式

OPEN“[デバイス名][ファイル名[. タイプ名]]”[FORモード]

AS[#]ファイル番号[LEN＝レコード長]

デバイス名　【条】CAS：，CRT：，GRP：，LPT：

【ディスク】A：，B：，C：，D：，E：，F：，G：，H：

【メモリ】MEM：

【省】CAS：

【ディスク】現在使用中のディスクドライブ。

ファイル名　【条】6字以内の文字列。

【メモリ】【ディスク】8字以内の文字列。

【省】ヌルストリング

（フロッピーディスク，メモリーディスク使用時は省略不可）

【メモリ】【ディスク】タイプ名　【条】3字以内の文字列。

【省】ヌルストリング。

モード　【条】OUTPUT，INPUT，【メモリ】【ディスク】APPEND

ファイル番号　【条】1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

【ディスク】レコード長　【条】1≦レコード長≦256。整数型定数，変数，配列変数，式。

【省】256

働きと使いかた

OPEN文は，指定したデバイス上のファイルに対して入出力を行うために，指定したファイル番号でファイルを開きます。指定できるデバイス中，CRT：，GRP：，LPT：は書き込み専用ですから，モードはOUTPUTしか指定できません。それに対し，CAS：は書き込み，読み取りとも可能ですのでOUTPUT，INPUTの両方を指定することができます。さらに，ディスクドライブ（A：，B：…）メモリーディスクでは追加モード（APPEND）も指定できます。

- ファイル番号は，すでにMAXFILES＝文で指定された，開くことのできるファイル数の最大値以内のいずれかの数値にします。

O

FORモードの省略

FORモードを省略すると，ランダムアクセスファイルに対して入出力を行うためにファイルを開くことを指示します。ランダムアクセスファイルは，ディスクでのみ使うことができます。

レコード長の指定

ランダムアクセスファイルに対するOPEN文の場合，最大レコード長を指定することができます。指定を省略すると，256に設定されます。

実行例

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 PSET (120,90)
40 PRINT #1,"ABC"
50 GOTO 50
```

グラフィックモード(SCREEN 2)のときにキャラクターを画面上に出力するプログラムです。

O

OUT（out）

指定したI/Oポートに1バイトのデータを出力する。　関INP

形　式

OUT ポート番号，式

ポート番号，式　条0≦番号<256。数値型定数，変数，配列変数，式。

働きと使いかた

I/Oポートに直接データを出力するコマンドです。

I/Oポートの割り当てについては275ページをご覧ください。

関数 PAD (pad)

コントローラー（タッチパッド，ライトペン，マウス，トラックボール）の状態を返す。

形式

PAD (N)

N 条0≦N＜20。数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

	Nの値	返される値の意味
タッチパッド	0または4	0：タッチパッドで画面上に触れていない −1：タッチパッドで画面上に触れている
	1または5	触れた位置のX座標
	2または6	触れた位置のY座標
	3または7	0：スイッチが押されていない −1：スイッチが押されている
ライトペン	8	0：ライトペンで画面上に触れていない −1：ライトペンで画面上に触れている
	9	触れた位置のX座標
	10	触れた位置のY座標
	11	0：ライトペンのスイッチが押されていない −1：ライトペンのスイッチが押されている
マウスまたはトラックボール	12または16	−1：入力要求に使う
	13または17	マウスまたはトラックボールのX座標
	14または18	マウスまたはトラックボールのY座標
	15または19	0

Nが0から3，または12から15のときはコントローラーAの状態を返し，4から7，または16から19のときはコントローラBの状態を返します。

PAD (12) またはPAD (16) は実行されたときのみ入力が行われ，PAD (13)，PAD (17)，PAD (14)，PAD (18) との時間間隔が長すぎるとカウンターがあふれて無意味な座標値が入力されます。

ボタンの状態を入力するためには，STRIG関数を使ってください。

実行例

```
100 SCREEN 5
110 FOR I=0 TO 7
120  A$=A$+CHR$(2^I-1)
130 NEXT
140 SPRITE$(0)=A$
150 D=PAD(12)
160 X=X+PAD(13):Y=Y+PAD(14)
170 PUT SPRITE 2,(X,Y),8,0
180 GOTO 150
```

P

PAINT (paint)

ペイント

境界線で囲まれた領域を塗りつぶす。

形　式

PAINT [STEP] (X座標, Y座標), [表示カラー], [境界線カラー]

X座標　条0≦X<513。数値型定数，変数，配列変数，式。

Y座標　条0≦Y<213。数値型定数，変数，配列変数，式。

表示カラー，境界線カラー

条SCREEN 2～7, 10のとき

0≦カラー<16。数値型定数，変数，配列変数，式。

SCREEN 8, 11, 12のとき

0≦カラー<256。数値型定数，変数，配列変数，式。

省現在のフォアグラウンドカラー。

働きと使いかた

(X座標，Y座標)で指定した位置を含み，指定された色の境界線で囲まれた領域を，表示カラーで塗りつぶします。

- 境界線が完全に閉じていないと，画面全体が塗りつぶされます。
- SCREEN 2，4のモードでは，表示カラーと境界線カラーが同じでないと境界線をはみ出してつぶされてしまいます。

STEPを指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標系でX，Yが求められます。

実行例

```
10 CLS
20 SCREEN 2
30 CIRCLE (70,70),40,10
40 PAINT (70,70),10,10
50 GOTO 50
```

SCREEN 2，4の場合，表示カラーと境界線カラーは同じ色を指定しなければならない。

```
10 CLS
20 SCREEN 3
30 CIRCLE (70,70),40,10
40 PAINT (70,70),8,10
50 GOTO 50
```

SCREEN3，5～8，10～12の場合，表示カラーと境界線カラーは違う色を指定できる。

関数 PDL（paddle）

（パドル）

パドルからの値を返す。

形式

PDL(N)

N　条1≦N＜13。数値型定数，変数，配列変数，式。

返される値：数値型定数，0≦値≦255。

働きと使いかた

パドルから得られる値を数値型データとして返します。Nが奇数の場合はコントローラー端子Aに接続されたパドルから，Nが偶数の場合はコントローラー端子Bに接続されたパドルからのデータが返されます。

関数 PEEK（peek）

（ピーク）

メモリーの指定した番地の内容を返す。　　反POKE

形式

PEEK(番地)

番地　条−32768≦番地＜65536。数値型定数，変数，配列変数，式。

　　負数の場合は，65536にその値を加えた値と同じ。

返される値：数値型，10進形式。

実行例

```
M=PEEK(50000)
```
……メモリーの50000番地の内容を変数Mに代入する。

P

PLAY (play)その1

プレイ

サブコマンドの指定どおりに音を出す。　　関SOUND, PLAY, BEEP

形　式

PLAY サブコマンド1[, サブコマンド2][, サブコマンド3]

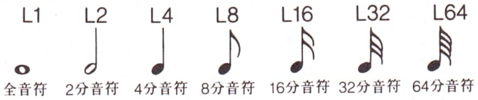
♪PLAY[#モード,]サブコマンド1[, サブコマンド2]. . . [, サブコマンド12]

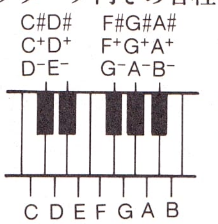
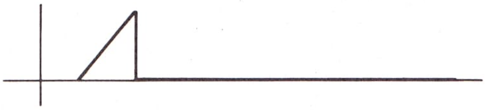
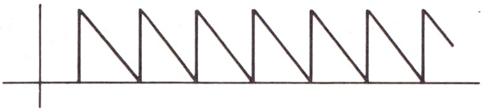
サブコマンド　条文字型定数または変数。

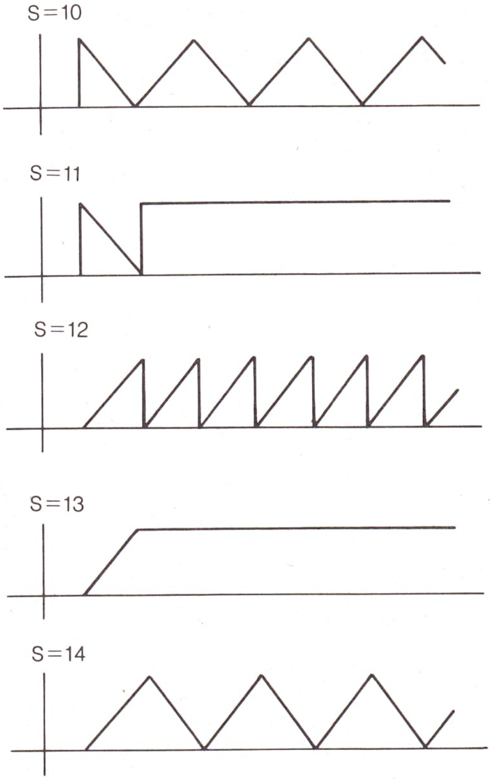
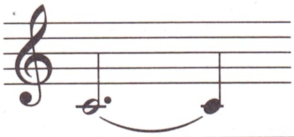
♪モード　条0≦モード≦3。

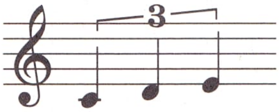
省0

サブコマンド

コマンド	条件	意　　味
Tn (tempo)	32≦n≦255 の整数	音楽の速さを指定。nの値は1分間に4分音符を何回数えるかを示す。 初期設定はT120。
On (octave)	1≦n≦8 の整数	8つのオクターブのうち何番目かを指定。 O4のとき，下図の範囲を演奏。 nの値が小さいほど下のオクターブ，大きいほど上のオクターブ 初期設定はO4
Ln (length)	1≦n≦64 の整数	音の長さを示す。  初期設定はL4
Nn (note)	0≦n≦96 の整数	音程を指定する。 N36はO4C N0は休符 nが1上がるごとに半音階上がる。

A—G An-Gn	1≦n≦64 の整数	指定したオクターブ内での音程を指定する。  半音は#（または＋）， －を付ける。 nで，その音の長さを指定することができる。 (C4はL4Cと同じ)。省略するとLnで指定した長さ。
Rn (rest)	1≦n≦64 の整数	休符を指定する。 R1 R2 R4 R8 R16 R32 R64 全休符 2分休符 4分休符 8分休符 16分休符 32分休符 64分休符
.		付点を表わす。 ひとつつけると長さが1.5倍になる。 C4.＝♩. R8.＝𝄾.
Vn (volume)	0≦n≦15 の整数	音量を指定する。 nが大きいほど音量が大きくなる。 初期設定はV8
Sn (shape)	0≦n≦15 の整数	音量変化のパターンを，次のうちどれかひとつ指定。 S＝0，1，2，3，9 S＝4，5，6，7，15  S＝8 

Sn (shape)	1≦n≦15 の整数	 初期設定はS1。 SコマンドとMサブコマンドの組み合わせでいろいろな音の出かたが決まる。
Mn (modulation)	1≦n≦65535 の整数	Sサブコマンドで指定したパターンの周期を決定する。 nの値が大きいほど周期が大きくなる。 初期設定はM255。
>		オクターブをひとつ上げる。
<		オクターブをひとつ下げる。
♪Qn	1≦n≦8	音の長さの割合。 初期設定はQ8。
& (Tie)		前後の同じ高さの音をつなぐ。（タイ） O4C2.&Cでは  を演奏。

♪{ }n	1≦n≦64	連符。n分音符を{ }の中の音程の個数で等分した音を演奏する。 O4{CDE}2では  を演奏。 nは省略するとLnで設定した値となる。
♪@n	1≦n≦63	n番の音色に切り換える。 音色番号については，CALL VOICEの項参照。
♪@Vn	0≦n≦127	音量を細く設定する。
@Wn	1≦n≦64	nで指定された長さだけ状態を継続する。 nは省略するとLnで設定した値となる。 PSG音源の場合，Rnと同じになる。
♪Yr,d		FM音源LSIのレジスターrにデータdを書き込む。

♪リズム音のサブコマンド

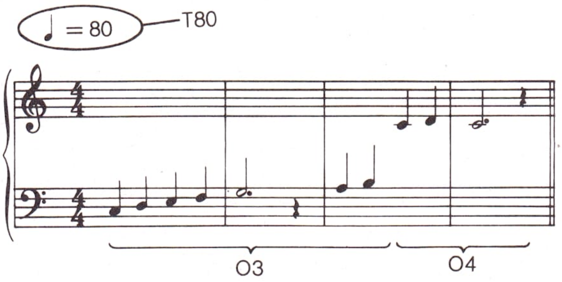
コマンド	条件	意味
B		バスドラム音を発生
S		スネアドラム音を発生
M		タムタム音を発生
C		シンバル音を発生
H		ハイハット音を発生
!		直前の楽器にアクセント
n	1≦n≦64	直前までの楽器音を発生しn分音符分待つ。
Vn	0≦n≦15	アクセントなしの楽器音の音量。初期設定はV8

@An	0≦n≦15	アクセントつきの楽器音の音量
Tn	32≦n≦255	テンポを決める。 初期設定はT120
@Vn	0≦n≦127	音量を細かく設定する。
Rn	1≦n≦64	休符を指定する。 初期設定はR4

働きと使いかた

```
PLAY "T80O3L4CDEFG2.RABO4CDC2."
```

この文により，次の楽譜どおりの音が演奏されます。



サブコマンド列を変数で表わす

```
M$="T80O3L4CDEFG2.RABO4CDC2."
PLAY M$
```

サブコマンド列をいったんM$という文字型変数に代入し、PLAY文ではサブコマンド列としてM$を指定しています。

サブコマンド列の一部を変数で表わす（X変数名 ;）

```
10 M$="CDEFG2.R"
20 PLAY "O4L4XM$;GAGAG2.R"
30 PLAY "XM$;ABO5CDC2."
```

文字型変数に代入したサブコマンド列を、PLAY文の“　”の中で使いたいときは、前にXを、後に ; をつけます。上の例では、M$に代入したサブコマンド列を2つのPLAY文中で使っています。

サブコマンドの中のnを変数で表わす（＝変数名；）

各サブコマンドで指定するnは，PLAY文中で定数にしても変数にしてもかまいません。変数で表わすときは，前に＝，後に；をつけます。

```
10 FOR I=1 TO 8
20 PLAY "O=I;CEG"
30 NEXT I
```

このプログラムは，PLAY “O1CEG” から始めて，PLAY “O8CEG” まで，8オクターブにわたる演奏をします。

和音の演奏

PLAY A$, B$, C$

のように，複数のサブコマンド列を同時に3つまで演奏させることも可能です。

```
10 A$="O4CDO3BO4E2R4"
20 B$="O4EFDG2R4"
30 C$="O4GAGO5C2R4"
40 PLAY A$,B$,C$
```

このプログラムは，次の楽譜を演奏します。

♪FM音源を使った演奏

FM音源を使用する場合，最初にCALL MUSIC文を実行してFM音源を使用可能な状態にします。このとき，FM音源のどのチャンネルからPLAY文のどのサブコマンド列を出力するかを指定します。また，リズム音を使用するかどうかも合わせて指定します。

CALL MUSIC文によるチャンネルの設定とPLAY文の各サブコマンド列との関係についてはCALL MUSICの項をご覧ください。

FM音源を使用する場合は，PLAY文でモードを必ず2または3にします。モードを0にしたり省略した場合は，PSGだけが使用できます。したがって，このときのサブコマンド列は最大3つまでとなります。

```
10 CALL MUSIC(0,0,3,2,1,1)
20 PLAY A$,B$,C$,D$
```

このプログラムでは，変数A$, B$, C$, D$にそれぞれサブコマンド列が代入されているとき，A$をチャンネル1, 2, 3から，B$をチャンネル4, 5から，C$をチャンネル6から，D$をチャンネル7から出力します。

♪リズム音の演奏

FM音源を使い，リズム音も演奏するときは，CALL MUSIC文のモードを1にするとともに，PLAY文ではメロディーを表わす最後のサブコマンド列の次にリズム音のサブコマンド列を書きます。

♪リズム音のサブコマンド列の例:

```
10 '  "BSH4H4S!4H4"
```

バスドラム，スネアドラム，ハイハットを鳴らして4分音符分待ち，ハイハットを鳴らして4分音符分待ち，スネアドラムをアクセントつきで鳴らして4分音符分待ち，ハイハットを鳴らして4分音符分待ちます。

```
10 CALL MUSIC(1,0,2,1,1,1,1)
20 PLAY A$,B$,C$,D$,E$,F$
```

このプログラムでは，変数A$～E$にメロディー演奏用のサブコマンド列が代入され，F$にリズム音用のサブコマンド列が代入されているとき，A$をチャンネル1，2から，B$をチャンネル3から，C$をチャンネル4から，D$をチャンネル5から，E$をチャンネル6から，F$をチャンネル7，8，9から出力します。

P

関数 PLAY（play）その2

音楽演奏中かどうか調べる。 関PLAY

形式

PLAY(N)

N 条0≦N＜4。数値型定数，変数，配列変数，式。

返される値：数値型

働きと使いかた

PLAY文では，同時に3つの音を演奏することができます。

PLAY A$，B$，C$

の場合，サブコマンドA$の音はチャンネル1から，B$の音はチャンネル2から，C$の音はチャンネル3から出力されます。

PLAY関数は，N＝1のときはチャンネル1の，N＝2のときはチャンネル2の，N＝3のときはチャンネル3の音楽データバッファ内にデータがあるかないかを調べる関数です。バッファにデータがあれば－1が，なければ0が返されます。N＝0のときはすべてのチャンネルのバッファの状態（0または1）のOR（論理和）をとって返します。つまり，どれかひとつでも－1になっていれば－1が返されることになります。

ご注意

FM音源使用時は，CALL PLAY文を使います。

P

関　数　POINT（point）

ポイント

グラフィックモードの画面で指定した位置の点のカラーコードを返す。

形　式

POINT（X座標，Y座標）

X座標，Y座標　条−32768≦座標＜32768。数値型定数，変数，配列変数，式。

返される値：数値型（指定位置が表示領域外のときは−1が返される）。

実行例

```
10 SCREEN 3
20 FOR I=1 TO 250
30 X=INT(RND(1)*255)
40 Y=INT(RND(1)*191)
50 PSET (X,Y),1
60 NEXT I
70 FOR Y=0 TO 191 STEP 4
80 FOR X=0 TO 255 STEP 4
90 C=POINT(X,Y)
100 IF C=4 THEN PSET (X,Y),15
110 NEXT X,Y
120 GOTO 120
```

行番号90で（X，Y）の位置のカラーコードを変数Cに代入し，行番号100でCが4（濃い青）なら同じ位置を白に変えます。

POKE (poke)

ポーク

メモリーの指定した番地にデータを書き込む。 反PEEK

形　式

POKE 番地，式

番地　条－32768≦番地＜65536。数値型定数，変数，配列変数，式。
　　　負数の場合は，65536にその値を加えた値と同じ。

式　条0≦式の値＜256。数値型定数，変数，配列変数，それらの式。

実行例

```
POKE 50000,255 …………メモリーの50000番地にデータ255を書き込む。
POKE &HD000,&HA8 ……メモリーのD000H番地にデータA8Hを書き込む。
```

関数 POS (position)

ポス

カーソルの位置のX座標を返す。

形　式

POS(X)

X　条任意の数値型定数，変数，配列変数，式。(ダミーの引数)

返される値：整数型。

実行例

```
10 INPUT A$
20 PRINT A$;:X=POS(X)
30 IF X>=5 THEN CLS
40 PRINT:GOTO 10
```

行番号20のX＝POS(X)により，XにはカーソルのX座標の値が代入されます。したがって，A$に5文字以上の文字列を入れると画面がクリアされます。

P

PRESEET (point reset)

グラフィックモードの画面に点を打つ。または消す。

形　式

PRESET ［STEP］（X座標，Y座標）［，カラー］

［,ロジカルオペレーション］

X座標，Y座標　条－32768≦座標＜32768。数値型定数，変数，配列変数，式。

カラー　条SCREEN 2～7, 10のとき

0≦カラー＜16。数値型定数，変数，配列変数，式。

SCREEN 8, 11, 12のとき

0≦カラー＜256。数値型定数，変数，配列変数，式。

省現在のバックグラウンドカラー。

ロジカルオペレーション　条SCREEN 5～8, 10～12のとき

PSET，PRESET，XOR，OR，AND

省PSET

働きと使いかた

カラーを省略して実行するとバックグラウンドカラーと同じ色の点を打ちます。したがって，指定した位置にバックグラウンドカラー以外で何かが描かれていれば，ちょうどその位置の点だけを消したように見えます。

- カラーを指定すると，PSETでカラーを指定した場合とまったく同じ働きをします。
- 実行例はPSETの項をご覧ください。

STEPを指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標系でX，Yが求められます。

P

PRINT（print）

プリント

テキスト画面上に数値データや文字データを表示する。　　　関LPRINT，PRINT USING

形　式

PRINT ［式］［セパレーター］［式］［セパレーター］［式］……

式　条数値型または文字型の定数，変数，配列変数，式。

セパレーター　条カンマ（,）またはセミコロン（；）

働きと使いかた

式（データ）の書きかた

数値型定数と数値型，文字型の変数はそのまま，文字型定数はダブルクォーテーション(”)で囲みます。

セパレーターの働き

カンマ（,）でデータを区切った場合は，データ間に14桁タブ機能によるスペースが入り，セミコロン（；）で区切った場合は次のデータが直後に続きます。

末尾にセパレーターを書かないと，データの表示後，改行が行われます。末尾にセパレーターを書くと，改行されず，次のPRINT文によるデータは同じ行に続きます。

数値データと符号

正負を表す符号は，“＋”の表示は省略されます。“－”はそのまま表示されます。(セパレーターが；の場合，正の数値型データを表示させると，データの間に符号のスペースとして2スペース入ります)

省略形

PRINTの代わりに，？をキーインしても結果は同じです。

実行例

```
10 A$="ABC":B$="DEF"
20 PRINT A$;B$
30 PRINT A$,B$
40 PRINT
50 PRINT "MSX"
60 PRINT +50,-50
70 ?"PERSONAL COMPUTER"
RUN
ABCDEF ―――――――――――――――――――――――― 行番号20の結果
ABC                 DEF ―――――――――― 行番号30の結果
                        ―――――――――― 行番号40の結果
MSX ――――――――――――――――――――――――――― 行番号の50の結果
 50                 -50 ―――――――――― 行番号60の結果
PERSONAL COMPUTER ―――――――――――――――― 行番号70の結果
```

P

PRINT USING (print using)

プリント ユージング

指定した書式でデータを画面へ出力する。 関PRINT, LPRINT USING

形　式

PRINT USING 書式記号；式［,式］…

式　条文字型，数値型の定数，変数，配列変数，式。

実行例

式の値が，書式記号で指定した形式で表示されます。

文字型データの書式記号

記号	表現形式と実行例

"!"	最初の1字を出力。
	`PRINT USING "!";"United","Nation"`
	`UN`

"&　　&"
n個のスペース
	n+2字分を出力。データがn+2字より少ない場合は残りの文字数だけスペースを挿入。
	`PRINT USING "&  &";"ABCDEF","GHI","JKLMN"`
	`ABCDGHI JKLM`

"@"	文字列全部を出力。
	`10 A$="North":B$="South"`
	`20 PRINT USING "@ Pole ";A$,B$`
	`RUN`
	`North Pole South Pole`

P

数値型データの書式記号

<table>
<tr><th>記号</th><th>表現形式と実行例</th></tr>
<tr><td>“#”</td><td>表示したい数字の桁数だけ#を書く。小数点は“.”。
<pre>PRINT USING "POINT:###.#";123.4
POINT:123.4</pre>■整数部分の桁数が指定した#の数より少ない場合,データは右づめで表示され,多い分はデータの前に“%”が付く。
<pre>10 PRINT USING "####";12
20 PRINT USING "####";12345
RUN
  12
%12345</pre>■数値データの小数部分の桁数が指定した#の数より少ない場合,“0”が追加され,逆の場合は四捨五入される。
<pre>10 PRINT USING "##.##";25.3
20 PRINT USING "##.##";25.345
RUN
25.30
25.35</pre>■数値データの“+”記号は無視,“−”記号は1桁として数える。
<pre>10 PRINT USING "###";+123
20 PRINT USING "###";-123
RUN
123
%-123</pre></td></tr>
<tr><td>“+”</td><td>数値データの前または後に,正数なら“+”を,負数なら“−”を付ける。
<pre>10 PRINT USING "+####";123,-123
20 PRINT USING "####+";123,-123
RUN
 +123 -123
 123+ 123-</pre></td></tr>
<tr><td>“−”</td><td>負の数値データの後に“−”を付ける。
<pre>PRINT USING "###-";123,-123
123 123-</pre></td></tr>
</table>

<table>
<tr><td>“＊＊”</td><td>数値データの前のスペースを“＊”で埋める。書式中の“＊”もひとつで1桁を表わす。<pre>
10 PRINT USING "**######";123
20 PRINT USING "**######";-234
RUN
*****123
****-234
</pre></td></tr>
<tr><td>“¥¥”</td><td>数値データの前に“¥”を付ける。書式中の“¥”もひとつで1桁に数えられる。<pre>
10 PRINT USING "¥¥###";1234
20 PRINT USING "+¥¥###";-1234
RUN
¥1234
-¥1234
</pre></td></tr>
<tr><td>“＊＊¥”</td><td>数値データの直前に“¥”を付け，それより前のスペースを“＊”で埋める。<pre>
PRINT USING "**¥###.##";12.34
***¥12.34
</pre></td></tr>
<tr><td>“,”</td><td>小数点の前のどこかに1つ指定すると，小数点から左に3桁ごとにカンマを入れて表示する。<pre>
PRINT USING "#,######.##";12345.67
  12,345.67
</pre></td></tr>
<tr><td>“＾＾＾＾”</td><td>数値データを浮動小数点で表示する。“＾＾＾＾”は指数部用のスペースに対応。<pre>
 PRINT USING "##.##^^^^";234.56
  2.35E+02
</pre></td></tr>
</table>

P

PRINT#（print number）

プリントナンバー

OPEN文で開かれたファイルにデータを書き込む。　関PRINT# USING, PRINT

形　式

PRINT#ファイル番号,［式］［セパレーター］［式］

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

式　条文字型，数値型の定数，変数，配列変数，式。

セパレーター　条カンマ（,）またはセミコロン（;）

働きと使いかた

OPEN文で開かれたファイルにデータを出力します。

実行例

```
10 OPEN "CAS:DATA" FOR OUTPUT AS #1 ――書き込み用に ファイルを開く
20 FOR I=0 TO 4
30 READ A$
40 PRINT #1,A$;",";   ――ファイルにデータを書き込む
50 NEXT I
60 CLOSE #1
70 DATA TOKYO,LONDON,PARIS,PEKING,NEW YORK
```

カセットテープにDATAというファイル名で行番号70に書かれたデータをひとつひとつ書き込んでいくプログラムです。

P

PRINT# USING (print number using)

(ルビ: プリントナンバー ユージング)

OPEN文で開かれたファイルに，指定した書式でデータを書き込む。

関PRINT#, PRINT USING

形　式

PRINT# ファイル番号，USING書式記号；式［,式］……

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

式　条文字型，数値型の定数，変数，配列変数，式。

働きと使いかた

ファイルにデータを出力するときに，書式を指定することができます。書式記号などについてはPRINT USINGの項を参照してください。

P

PSET（point set）

ピーセット

グラフィックモードの画面に点を打つ。

形　式

PSET [STEP]（X座標，Y座標），[カラー]，[ロジカルオペレーション]

X座標，Y座標　条－32768≦座標＜32768。数値型定数，変数，配列変数，式。

カラー　条SCREEN 2～7, 10のとき

0≦カラー＜16。数値型定数，変数，配列変数，式。

SCREEN 8, 11, 12のとき

0≦カラー＜256。数値型定数，変数，配列変数，式。

省現在のフォアグラウンドカラー。

ロジカルオペレーション　条SCREEN 5～8, 10～12のとき

PSET，PRESET，XOR，OR，AND

省PSET

働きと使いかた

STEP文を指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標系でX，Yが求められます。

実行例

```
10 SCREEN 2
20 FOR X=0 TO 255
30 PSET (X+1,100)      ――点を描く
40 PRESET (X,100)      ――ひとつ前に描いた点を消す
50 NEXT X
```

P

PUT (put)

ランダムアクセスファイルへ，FIELD文によって定義されたレコードを出力する。反GET

形　式

PUT ［#］ファイル番号［,レコード番号］

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

レコード番号　条1≦レコード番号＜65536。数値型定数，変数，配列変数，式。
省直前のGET文，PUT文で扱ったレコード番号＋1。

働きと使いかた

```
PUT #1,3
```

を実行すると，FIELD文，LSET/RSET文により準備されたランダムアクセスファイル用のレコードが第3番目のレコードとしてファイルに出力されます。

実行例

```
10 OPEN "A:TEST.DAT" AS #1
20 FIELD #1,2 AS CODE$,15 AS NAM$,10 AS  TEL$
30 READ A%,B$,C$
40 LSET CODE$=MKI$(A%)
50 LSET NAM$=B$
60 LSET TEL$=C$
70 PUT #1,1
80 CLOSE #1
90 DATA 100,PAUL BROWN,123-4567
100 END
```

P

PUT KANJI (put kanji)

（プット カンジ）

漢字を表示する。

形　式

PUT KANJI [(X座標, Y座標)], 漢字コード, [カラー], [ロジカルオペレーション], [モード]

X座標, Y座標　条−32768≦座標<32768。数値型定数, 変数, 配列変数, 式。
省直前のグラフィック命令で最後に指定された位置。

漢字コード　条&H2121≦コード≦&H4F53, JIS漢字コード

カラー　条SCREEN 5〜7, 10のとき
0≦カラー<16。数値型定数, 変数, 配列変数, 式。
SCREEN 8, 11, 12のとき
0≦カラー<256。数値型定数, 変数, 配列変数, 式。
省フォアグラウンドカラー。

ロジカルオペレーション　条PSET, PRESET, XOR, OR, AND
省PSET

モード　条0≦モード<3。数値型定数, 変数, 配列変数, 式。
省0

働きと使いかた

漢字の表示には漢字ROMが必要です。SCREEN 5〜8およびSCREEN 10〜12のとき有効です。

モード	機　　能
0	16×16ドットで普通に表示する。
1	偶数番目の線だけを縦8ドットで表示する。
2	奇数番目の線だけを縦8ドットで表示する。

ご注意

- JIS第二水準の漢字を表示するには, JIS第二水準漢字ROMが必要です。
- SCREEN 11および12のモードでは, 漢字の周囲に色バケを起こします。

P

PUT SPRITE (put sprite)

指定したスプライトパターンを，指定したスプライト面の任意の位置に表示する。

関SPRITE ON，SPRITE OFF，SPRITE STOP，SPRITE$

形　式

PUT SPRITE スプライト面番号, [[STEP] (X座標, Y座標)], [カラー], [スプライト番号]

スプライト面番号　条0≦番号<32。数値型定数，変数，配列変数，式。

X座標　条−32≦X<256。数値型定数，変数，配列変数，式。

Y座標　条−32≦Y<212。数値型定数，変数，配列変数，式。

カラー　条0≦カラー<16。数値型定数，変数，配列変数，式。
省現在のフォアグラウンドカラー。

スプライト番号　条8×8ドットのとき…0≦番号<256。数値型定数，変数，配列変数，式。
16×16ドットのとき0≦番号<64。数値型定数，変数，配列変数，式。
省スプライト面番号と同じ。

働きと使いかた

STEPを指定すると，直前のグラフィック文で最後に指定された点を原点とする新しい座標系でX，Yが求められます。

SCREEN 8, 11, 12モードでのカラー

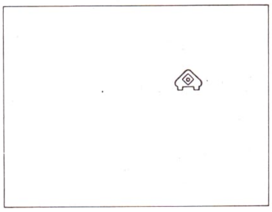
SCREEN 8, 11, 12モードでも，PUT SPRITE文で指定できるカラーは0から15までで，その色は以下のように固定されています。

カラー指定	色	カラー指定	色	カラー指定	色	カラー指定	色
0	黒	4	暗い緑	8	肌色	12	緑
1	暗い青	5	暗い水色	9	青	13	水色
2	暗い赤	6	暗い黄	10	赤	14	黄
3	暗い紫	7	灰色	11	紫	15	白

P

実行例

```
10 SCREEN 2
20 SPRITE$(1)=CHR$(&H18)+CHR$(&H3C)+CHR$
(&H66)+CHR$(&HDB)+CHR$(&HE7)+CHR$(&H7E)+
CHR$(&H24)+CHR$(&H42)
30 X=0:Y=0:DX=1:DY=1
40 PUT SPRITE 0,(X,Y),,1
50 X=X+DX:Y=Y+DY
60 IF X>250 OR X<0 THEN DX=-DX
70 IF Y>190 OR Y<0 THEN DY=-DY
80 GOTO 40
```



行番号20でUFOの形をスプライトパターンとして定義し，スプライト番号１に割り当てています。そのスプライトパターンを画面に表示させているのが行番号40のPUT SPRITE文です。表示させるスプライト面は0番，表示カラーは省略してあるので，すでに設定されたフォアグラウンドカラーと同じになります。表示位置を指定するX，Yは値が次々に変わる変数で，これによりUFOのパターンが画面上を飛び回るように見えます。

P

READ (read)

リード

DATA文中のデータを読みとる。　　　関DATA, RESTORE

形　式

READ 変数 [, 変数] [, 変数] …

変数　条数値型または文字型。

働きと使いかた

プログラムの中で最も若い行番号を持つDATA文の先頭から順にデータを読みとり, READ文中の変数に順に代入します。

- 1つのREAD文中に複数個の数値型や文字型の変数を並べるときは, カンマ(,)で区切ります。
- 対応する変数の型とデータの型は一致していなくてはなりません。

```
10 READ A,B,C,D$,E$
20 PRINT A,B,C,D$,E$
100 DATA 5,10,20,ABC,XYZ
```

- 1つのプログラム中に複数のREAD文がある場合, 2番目のREAD文は前のREAD文が読み終えた次のデータから読み取りを開始します。
- RESTORE文を実行すると, 次に実行されるREAD文の読み取りはRESTORE文で指定した行番号以降で最も若い行のDATA文に戻ります。

実行例

```
10 READ A,B,C
20 READ D$,E$
30 PRINT A;B;C;D$;E$
100 DATA 10,20,30,ABC,DEF
RUN
 10  20  30 ABCDEF
```

R

REM (remark)

レム

プログラムに注釈文をつける。

形　式

REM 注釈文

働きと使いかた

REM文は，プログラムのリストをとったときに見やすいように，注釈文をつけることを可能にします。

実行例

```
10 REM MUSIC
20 PLAY "T60CEGEC1"
```

リストをとると，REM文はそのまま表示されますが，実行はREM文をとばして行われます。

```
10 'MUSIC
20 PLAY "T60CEGEC1"
```

REMのかわりにアポストロフィー（'）を使うことができます。

```
10 PRINT "MSX":REM Output
20 PRINT "PERSONAL COMPUTER" 'Output
RUN
MSX
PERSONAL COMPUTER
```

他の文のうしろにREM文を付けるときは，コロン(：)が必要ですが，'を使えばコロンを省略できます。

R

RENUM（renumber）

リナム

プログラムの行番号をつけ直す。

形　式

RENUM［新先頭行番号］,［旧先頭行番号］,［増分］

新先頭行番号　条0≦番号≦65529。整数型定数。
　省10

旧先頭行番号　条0≦番号≦65529。整数型定数。
　省実行前の最も小さい行番号。

増分　条0≦増分≦65529。整数型定数。
　省10

働きと使いかた

プログラムの修正後，行番号をきれいにつけ直したい場合などに使用します。

- GOTO文やGOSUB文中での飛び先の行番号も，RENUM文の実行によって正しくつけ直されます。
  ただし，GOTO文などの飛び先として指定された行がない場合，RENUMを実行してもGOTO文などの飛び先の行番号はそのまま変わらず，エラーとなります。

実行例

```
RENUM              ―― 全行を行番号10から増分10でつけ直す。

RENUM 100,,100     ―― 全行を行番号100から増分100でつけ直す。

RENUM 100          ―― 全行を行番号100から増分10でつけ直す。

RENUM 100,38,20    ―― 元のプログラムの行番号38以降を，行番号100から20おきにつけ直す。
LIST               ―― LISTを実行
15 FOR I=0 TO 10
20 A=A+1
23 PRINT A
35 NEXT I
Ok
RENUM              ―― RENUMを実行
Ok

LIST               ―― もう1度LISTを実行
10 FOR I=0 TO 10
20 A=A+1
30 PRINT A
40 NEXT I
```

R

RESTORE (restore)

リストア

READ文が読み取るDATA文を指定する。 関DATA, READ

形　式

RESTORE [行番号]

行番号　条0≦番号≦65529。整数型定数。

省最も行番号の小さいDATA文。

働きと使いかた

RESTORE文は，同じデータを複数回読みとる必要がある場合に使用します。
RESTORE文を実行すると，次のREAD文は，RESTORE文によって指定された行番号以降で，最も行番号の若いDATA文の先頭データから読み取りを開始します。

実行例

```
10 READ A,B,C
20 READ D,E,F
30 RESTORE 110
40 READ G,H,I
50 PRINT A;B;C;D;E;F;G;H;I
100 DATA 10,20,30
110 DATA 40,50,60
RUN
 10  20  30  40  50  60  40  50  60
```

R

RESUME (resume)
（レジューム）

エラー処理ルーチンの実行後，メインプログラムに実行を戻す。

関ON ERROR GOTO，RETURN

形　式

RESUME [{ 0 / 行番号 / NEXT }]

行番号　条0≦番号≦65529。整数型定数。

省エラーが生じた行。

働きと使いかた

```
RESUME 0  または RESUME ——エラーが生じた文に戻る
RESUME 100 ——行番号100に戻る
```

(「ON ERROR GOTO」の項のプログラム例参照)

関　数　RIGHT$ (right dollar)
（ライトダラー）

文字データの右から任意の数の文字を文字データとして返す。　関LEFT$，MID$

形　式

RIGHT$(X$,N)

X$　条文字型定数，変数，配列変数，式。

N　条0≦N<256。数値型定数，変数，配列変数，式。

返される値：文字型

実行例

```
PRINT RIGHT$("I LOVE TOKYO",5)
TOKYO
PRINT RIGHT$("I LOVE TOKYO",5.3)   } Nが整数値でない場合，
TOKYO                              } 小数点以下が切り捨てられる。
PRINT RIGHT$("I LOVE TOKYO",0)
                                   } Nが0のときはヌルストリングを返す。
Ok
```

ご注意

日本語入力モードで同様の操作を行うには，CALL KMID文を使います。ただし，働きはLEFT$関数と同じになります。

R

関数 RND（random）

（ランド）

0以上1未満の乱数を返す。

形式

RND(X)

X 条数値型定数，変数，配列変数，式。

返される値：数値型。

働きと使いかた

0<Xのとき

常に一定の系列の乱数を発生します。

```
10 FOR N=1 TO 10
20 PRINT RND(1)
30 NEXT N
RUN
 .59521943994623
 .10658628050158
 .76597651772823
 .57756392935958
 .73474759503023
 .18426812909758
 .37075377905223
 .94954151651558
 .63799556899423
 .47041117641358
```

X=0のとき

1つ前に発生したのと同じ値を返します。

```
10 PRINT RND(1)
20 PRINT RND(0)
30 PRINT RND(-1)
40 PRINT RND(0)
RUN
 .59521943994623
 .59521943994623
 .04389820420821
 .04389820420821
```

X<0のとき

Xの値に対応する発生系列をつくり，それ以降のRND関数はその系列で乱数を発生します。

```
10 PRINT RND(-1)
20 FOR N=1 TO 10
30 PRINT RND(N)
40 NEXT N
RUN
 .04389820420821
 .0962486816692
 .21069655852301
 .3265173630504
 .47775124336581
 .3409147084636
 .12971184081661
 .0977770174288
 .35157860175541
 .835389696666
 .63902641386221
```

RSET（right set）

（アールセット）

ランダムアクセスファイルのレコードにデータを右づめで書き込む。　関LSET

形　式

RSET 文字型変数＝文字式

文字型変数　条文字型の変数

文字式　条文字型定数，変数，配列変数，式。

働きと使いかた

PUT文でランダムアクセスファイルにデータを書き込む準備として，FIELD文で定義したレコード内の各変数にデータをセットします。RSET文は，データを右づめでセットします。このとき，FIELD文で指定した変数の長さよりもデータが短ければ，余白はスペースで埋められ，データのほうが長ければ文字データの右側が無視されます。

● 数値データをセットするときは，MKI$，MKS$，MKD$の各関数により，いったん文字型に変換します。

```
RSET A$=X$
```
……文字データX＄を，レコード内の変数A$にセット

```
RSET B$=MKS$(N)
```
……数値データNを文字型に変換してレコード内の変数B＄にセット

R

RUN (run)

（ラン）

1. 指定した行から，プログラムを実行する。

2. ディスクやメモリーディスクのファイルからプログラムをロードして実行する。

形　式

1. RUN［行番号］

行番号　条0≦番号≦65529。整数型定数。

　　　　省先頭行から実行。

2. RUN "［ドライブ名］ファイル名［.タイプ名］"［, R］

ドライブ名　条A：,B：,C：,D：,E：,F：,G：,H：,　MEM：

　　　　　　省現在使用中のディスクドライブ。

ファイル名　条8字以内の文字列。

タイプ名　　条3字以内の文字列。

　　　　　　省ヌルストリング。

Rオプション　省データファイルを閉じる。

働きと使いかた

形式1の場合

RUN文が実行されると，すべての数値変数が0に，文字変数がヌルストリングに，配列変数が未定義状態になってプログラムが実行されます。

プログラムの実行が終了するとコマンド待ち状態に戻ります。

- プログラムの実行を一時停止するには，[STOP]を押します。もう一度押すと実行が再開します。

  実行を中断するときは[CTRL]と[STOP]を押します。この場合は，CONTコマンドで再開できます。

形式2の場合

このRUN文が実行されると，すべての数値変数は0に，文字変数はヌルストリングに，配列変数は未定義状態になるとともに，ファイルはすべて閉じられます。

次に，指定したディスクドライブから指定したファイル名，タイプ名のプログラムをロードして実行します。

Rオプションをつけると，すべてのデータファイルは開かれたままになります。

SAVE (save)

セーブ

BASICプログラムを指定したデバイスにセーブする。　反LOAD　関CSAVE

形　式

SAVE "[デバイス名][ファイル名[.タイプ名]]"[, A]

デバイス名　系CAS:, CRT:, GRP:, LPT:

MEM:

A:, B:, C:, D:, E:, F:, G:, H:

省CAS:

現在使用中のディスクドライブ。

ファイル名　系6字以内の文字列。

8字以内の文字列。

省ヌルストリング

(フロッピーディスク, メモリーディスク使用時は省略不可)。

タイプ名　系3字以内の文字列。

省ヌルストリング。

Aオプション　省内部表現形式でセーブ。

働きと使いかた

デバイス名にCAS:を指定すると, メモリー上のBASICプログラムがアスキー形式でカセットテープにセーブされます。

デバイス名にディスクドライブ(A:, B:など)を指定した場合, Aオプションをつけるとアスキー形式で, Aオプションをつけないと内部表現形式でプログラムがセーブされます。

ご注意

カセットテープに, プログラムを内部表現形式でセーブするときは, CSAVE文を使います。

実行例

```
SAVE "CAS:PROG2"   ……カセットテープにアスキー形式でプログラムをセーブ。
SAVE "PROG2",A     ………現在使用中のディスクにアスキー形式でプログラムをセーブ。
```

- MERGE文でメモリー上のプログラムと合併させたいプログラムは, 必ずアスキー形式でセーブしてください。
- メモリーディスクへのセーブは常にアスキー形式でおこなわれます。

S

スクリーン
SCREEN (screen)

画面表示モード，スプライトの大きさ，キー音の有無，カセットインターフェースのボーレート，インターレースモードを設定し，プリンターの種類を選択する。

形　式

SCREEN[モード]，[スプライトの大きさ]，[キークリックスイッチ]，[ボーレート]，[プリンタータイプ]，[インターレースモード]

モード　条0≦モード≦8，10≦モード≦12。数値型定数，変数，配列変数，式。
　省現在のモード。

スプライトの大きさ　条0≦大きさ<4。数値型定数，変数，配列変数，式。
　省現在の大きさ。

キークリックスイッチ　条0≦スイッチ<256。数値型定数，変数，配列変数，式。
　省現在の状態。

ボーレート　条1≦ボーレート<3。数値型定数，変数，配列変数，式。
　省現在のボーレート。

プリンタータイプ　条0≦プリンタータイプ<256。数値型定数，変数，配列，式。
　省現在のプリンタータイプ。

インターレースモード　条0≦モード<4。数値型定数，変数，配列変数，式。
　省現在のモード。

S

働きと使いかた

モード

	指定値	モード
*	0	40字×24行，または80字×24行テキストモード
	1	32字×24行テキストモード
	2	ハイレゾリューションモード
	3	マルチカラーモード
	4	スプライト機能強化型ハイレゾリューションモード
	5	256×212ドット　16色　ビットマップディスプレイモード
	6	512×212ドット　4色　ビットマップディスプレイモード
	7	512×212ドット　16色　ビットマップディスプレイモード
	8	256×212ドット　256色ビットマップディスプレイモード
**	10	256×212ドット　16色ビットマップディスプレイモード
**	11	256×212ドット　256色ビットマップディスプレイモード
**	12	256×212ドット　256色ビットマップディスプレイモード

* WIDTH文で指定された幅が40以下ならば40字×24行テキストモードが選ばれ，41以上ならば80字×24行テキストモードが選ばれる。

** 詳しくは13ページの「YJK方式」を参照してください。

SCREEN 0～SCREEN 8を実行すると，画面表示は初期状態に戻ります。SCREEN 10, 11, 12の場合，あるモードからあるモードへ切り換えても次の場合は表示は初期化されません。

SCREEN 10から11へ
SCREEN 10から12へ
SCREEN 11から10へ
SCREEN 11から12へ
SCREEN 12から10へ（アトリビュートビットは0にセットされます。）
SCREEN 12から11へ（アトリビュートビットは0にセットされます。）

SCREENモードの違いによる詳しい機能については，11ページをご覧ください。

スプライトの大きさ

指定値	大きさ
0	8×8ドット標準
1	8×8ドット拡大
2	16×16ドット標準
3	16×16ドット拡大

キークリックスイッチ

	指定値	キーを打ったときの音
	0	なし
*	0以外	あり

*1から255までの範囲。

ボーレート

指定値	ボーレート*
1	1200ボー
2	2400ボー

*カセットインターフェースのボーレート。

S

プリンタータイプ

	指定値	プリンター
	0	MSXプリンター**
*	0以外	MSXプリンター以外***

*1から255までの範囲。

**グラフィックキャラクターとひらがなのフォントを内蔵し，MSXパーソナルコンピューターに対応したプリンター。

***MSXプリンター以外では，グラフィックキャラクターはスペースに，ひらがなはカタカナに変換される。

インターレースモード

	指定値	インターレース
	0	通常
	1	インターレース
*	2	Even（偶数）／odd（奇数）
*	3	Even／odd インターレース

*ディスプレイページは奇数でなくてはならない。ディスプレイページとディスプレイページより1つ小さい番号のページが交互に表示される。

初期値と指定の省略

指定を省略するとすでに設定されているモードのまま変わりませんが，すべてを省略してSCREENを実行することはできません。

MSX_{2+}-BASIC起動時の初期状態はSET SCREENにより変えることができますが，出荷時の状態は以下のとおりです。

モード：32字×24行テキストモード（WIDTH29）

スプライトの大きさ：8×8ドット標準

キークリックスイッチ：キー音あり

ボーレート：1200ボー

プリンタータイプ：MSXプリンター

インターレースモード：通常

S

実行例

```
10 SCREEN 0,,1
```
……40字×24行テキストモード，キー音なし

```
10 SCREEN ,,,2
```
……ボーレートを2400ボーに

```
10 SCREEN 2,3
```
……ハイレゾリューションスプライトは16×16ドット拡大

```
10 SCREEN 2
20 FOR I=0 TO 255
30 PSET (I,100)
40 NEXT I
50 GOTO 50
```

プログラムの実行が終了すると，画面はテキストモード(SCREEN 0または1)に戻ります。したがって，グラフィックモードの状態を続けたいときは，上の例の行番号50のように，プログラムを実行状態のままにする方法があります。
実行を止めるには，CTRLとSTOPを同時に押します。

S

SET ADJUST (set adjust)

セット　アジャスト

画面の表示位置を変える。

形　式

SET ADJUST（X座標，Y座標）

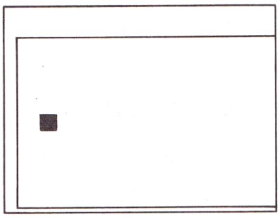
X座標，Y座標　条−8<座標<9。数値型定数，変数，配列変数，式。

働きと使いかた

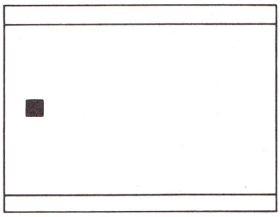
正の値を入れるとX方向では右に，Y方向では下に画面表示が移動します。初期値はX座標，Y座標とも0です。

実行例

```
SET ADJUST (8,8)
```



```
SET ADJUST (0,0)
```



セット　ビープ

SET BEEP（set beep）

BEEP音を選択する

形　式

SET BEEP［音色］,［音量］

音色　条1≦音色<5。数値型定数，変数，配列変数，式。

音量　条1≦音色<5。数値型定数，変数，配列変数，式。

働きと使いかた

音色は次のようになっています。

1　ピッ

2　ポーン

3　ピンポーン

4　パピポ

音量は１を指定すると最小となり，4を指定すると最大になります。

SET DATE (set date)

（セット デート）

内蔵の時計に日付けを設定する。　反GET DATE　関SET TIME, GET TIME

形　式

SET DATE T$ [, A]

T$　条文字型定数，変数，配列変数，式。

A　省現在の年月日。

働きと使いかた

文字型定数で，設定する日付けを指定します。日付けデータの形式はYY／MM／DDでYYは西暦の下2ケタ，MMは月，DDは日を表します。

Aを設定するとアラームの日付けを設定することができます。設定したアラームの日に何が起きるかについてはお使いになる機器の取扱説明書をご覧ください。(機器によっては機能しないものもあります。)

アラームの日付けを設定する場合，先に日付けをSET DATE文で設定してからSET TIME X$, Aを実行すると先に設定しておいたアラームの日付けが消されます。アラームの日付けと時刻の両方を設定する場合には，まずSET TIMEを実行し，次にSET DATEを実行してください。

実行例

```
10 T$="85/01/01"
20 SET DATE T$
30 PRINT T$
```

S

SET PAGE (set page)

ビデオRAMの複数のページを使用するとき，

データを書き込むページと画面に表示するページを指定する。

形　式

SET PAGE [ディスプレイページ] [, アクティブページ]

ディスプレイページ，アクティブページ

条 0≦ページ<4。数値型定数，変数，配列変数，式。

働きと使いかた

SCREEN 5〜8およびSCREEN 10〜12のとき有効です。

アクティブページとはデータが書き込まれるページ，ディスプレイページとはモニターテレビに表示されるページです。

スクリーンモード	ページ番号
5	0〜3
6	0〜3
7	0〜1
8	0〜1
10	0〜1
11	0〜1
12	0〜1

S

実行例

```
100 SCREEN 5
110 SET PAGE 0,1:CLS
120 LINE(70,50)-(170,150),,BF
130 SET PAGE 0,0
140 LINE(60,60)-(180,140),,BF
150 DP=0:AP=1
160 SET PAGE DP,AP:SWAP DP,AP
170 FOR I=0 TO 100:NEXT I
180 GOTO 160
```

SET PASSWORD (set password)

（セット　パスワード）

システムのパスワードを設定します。 関SET PROMPT, SET TITLE

形　式

SET PASSWORD "パスワード"

パスワード 条255字までの文字式。

働きと使いかた

パスワードを設定すると,システムを立ち上げるときにパスワードの入力が要求されます。正しいパスワードを入力すると正常にシステムが起動されますが，誤りがあるとシステムは起動されずパスワードの入力の要求が繰り返されます。

パスワードを設定しておくことで他人がシステムを起動するのを防ぐことができます。

パスワードを忘れたときは[GRAPH]キーと[STOP]キーを押しながら,[RESET]キーを押すと,システムを立ち上げることができます。

SET PASSWORD，SET TITEL，SET PROMPTの指定はそのうちのどれか1つしか有効ではありません。従って，SET TITLEまたはSET PROMPTを実行し，タイトルかプロンプトを指定するとパスワード機能は解除されます。

SET PROMPT (set prompt)

（セット　プロンプト）

BASICがコマンド待ち状態になったとき表示されるプロンプトを指定する。

関SET PASSWORD, SET TITLE

形　式

SET PROMPT "プロンプト"

プロンプト 条6字以内の文字列

働きと使いかた

SET PROMPT，SET PASSWORD，SET TITLEの指定はそのうちのどれか1つしか有効ではありません。従って，SET TITLEまたはSET PASSWORDを実行し，タイトルまたはパスワードを指定するとプロンプトはOKと表示されます。

SET SCREEN (set screen)

セット　スクリーン

現在設定されているSCREEN，COLOR，WIDTH，KEY ON/OFF文の指定値をシステム起動時の状態として登録する。

形　式

SET SCREEN

働きと使いかた

現在の画面の次の指定値がシステム起動時の状態として登録されます。

画面モード（テキストモードでの値）
表示幅（テキストモードでの値）
フォアグラウンドの色
バックグラウンドの色
ボーダーエリアの色
ファンクションキースイッチ
キークリックスイッチ
プリンターオプション
カセットボーレート
ディスプレイモード

SET SCROLL (set scroll)

セットスクロール

現在のディスプレイページをスクロールする。

関SET PAGE

形　式

SET SCROLL[X方向][,Y方向][,マスク][,ページ]

X方向　条0≦X方向≦511。数値型定数，変数，配列変数，式。
　省0。

Y方向　条0≦Y方向≦255。数値型定数，変数，配列変数，式。
　省0。

マスク　条0または1。数値型定数，変数，配列変数，式。
　省0。

ページ　条0または1。数値型定数，変数，配列変数，式。
　省0。

働きと使いかた

SCREEN0以外のモードの場合

X方向で，横方向（右から左）にスクロールさせる量を，Y方向でたて方向（下から上）にスクロールさせる量を，それぞれドット単位で指定します。

X,Y方向のスクロール量は，初期状態からの絶対的な値です。したがって，すでにSET SCROLL文でスクロールした画面をさらに同じXとYの値を使ってSET SCROLL文でスクロールさせることはできません。

また，スクロール終了後は画面のX座標，Y座標がスクロールの結果によって表示画面内で移動します。あらたにSCREEN文でスクリーンモードを設定し直すと，(0,0)の位置が画面の左上隅に戻ります。

SCREEN0モードの場合

X方向の値は0から7の範囲で指定可能です。0（または省略）のときは横方向にスクロールしません。1から7を指定すると，1で7ドット，2で6ドット，以下同様に7で1ドット分右にスクロールします。

Y方向の値は0から7の範囲で指定可能です。0（または省略）のときはたて方向にスクロールしません。1から7を指定すると，1で1ドット，2で2ドット，以下同様に7で7ドット分上にスクロールします。ただし，たて方向のスクロールを指定した場合，文字の位置自体は変化せず，同じ行の中でスクロールします。

また，スクロール終了後は表示画面が全体的にX座標方向に移動しています(横方向にスクロールさせた場合)。また，Y座標方向にも各行の範囲内で文字パターンがスクロールしています(たて方向にスクロールさせた場合)。あらたにSCREEN文でスクリーンモードを設定し直すと，初期状態に戻ります。

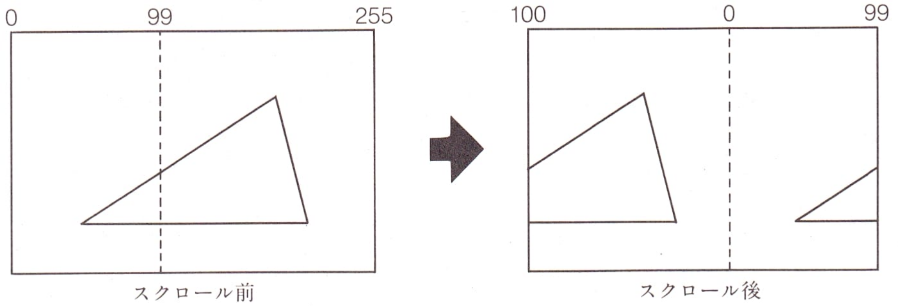
マスクの指定

マスクを1にすると，スクロール画面の左端8ドット分が現在のボーダー色でおおわれます。0にするか省略すると，ボーダー色でおおわれません。

横方向2ページにわたるスクロール

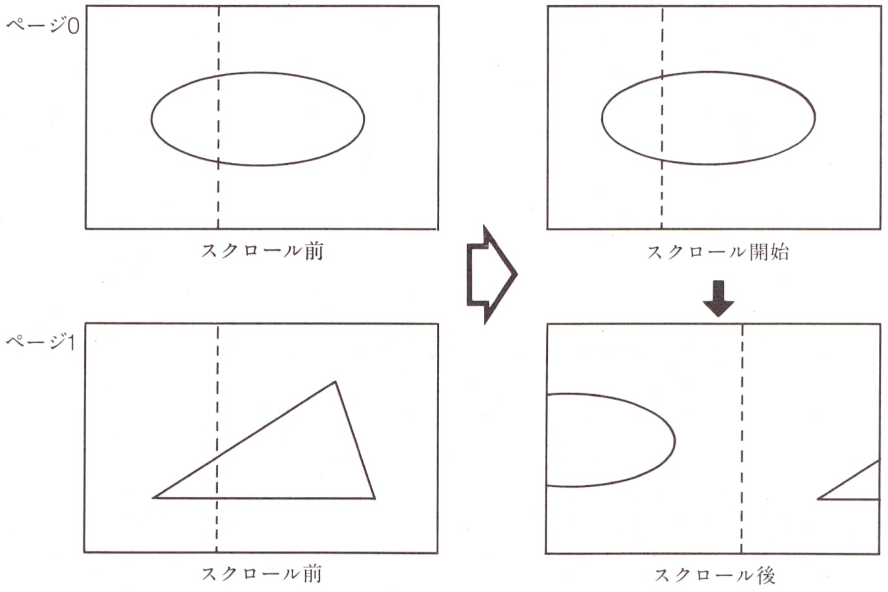
ページで，横方向のスクロール時に2ページ連続でスクロールするかどうかを指定します。ページが0または省略されたときは，同一ページでのスクロールとなります。

たとえば，SCREEN5のモードで横に100ドットスクロールした場合，スクロール前とスクロール後の画面は次のようになります。



S

ページが1のときは，2ページにわたるスクロールになります。このときスクロール開始直前のディスプレイページは奇数ページ（1または3）でなければなりません。
たとえば，SCREEN5のモードでページ0とページ1を使って横に100ドットスクロールした場合，スクロール前とスクロール後の画面は次のようになります。



実行例

```
10 SCREEN 5
20 SET PAGE 1,1
30 LINE (10,10)-(100,100),7,BF
40 FOR J=0 TO 500:NEXT J
50 FOR I=0 TO 255
60 SET SCROLL I,I,1,0
70 NEXT I
80 FOR I=255 TO 0 STEP -1
90 SET SCROLL I,I,1,0
100 NEXT I
110 GOTO 50
```

SET SCROLL文によるスクロールの効果を確かめるためのプログラムです。

SET TIME (set time)

時計に時刻を設定する。　　反GET TIME　関TIME, SET DATE, GET DATE

形　式

SET TIME X$ [, A]

X$　条文字型定数，変数，配列変数，式。

A　省現在時刻。

働きと使いかた

文字型定数で，設定する時刻を指定します。時刻データの形式はHH：MM：SSでHHは時，MMは分，SSは秒を表します。

,Aを指定するとアラームの時刻を設定することができます。設定したアラームの時刻に何が起きるかについてはお使いになる機器の取扱説明書をご覧ください。(機器によっては機能しないものもあります。)

アラーム時刻を設定する場合，先に日付けをSET DATE文で設定してからSET TIME X$, Aを実行すると先に設定しておいたアラームの日付けが消されます。アラームの日付けと時刻の両方を設定する場合には，まずSET TIMEを実行し，次にSET DATEを実行してください。

実行例

```
10 X$="11:00:00"
20 SET TIME X$
30 PRINT X$
```

S

SET TITLE (set title)

初期画面に表示されるタイトルを指定する。　関SET PASSWORD, SET PROMPT

形　式

SET TITLE [“タイトル”], [カラー]

文字列　条6字以内の文字列。
　　　　省現在の文字列。
カラー　条1≦カラー<5。数値型定数, 変数, 配列変数, 式。
　　　　省現在の色。

働きと使いかた

初期画面に希望のタイトルを付けることができます。

文字列で6字指定すると初期画面で動作が止まり任意のキーが押されるのを待ちます。

SET TITLE, SET PROMPT, SET PASSWORDの指定は常にそのうちのどれか1つしか有効ではありません。従って, SET PROMPTまたはSET PASSWORDを実行し, プロンプトまたはパスワードを指定するとタイトルは消されます。

実行例

```
SET TITLE "SONY",1
```

SET VIDEO（set video）

スーパーインポーズ，オーディオミキシング，その他のモードを設定する。

形　式

SET VIDEO［モード］，［輝度］，［カラーバス］，［同期］，［音声］，［ビデオ入力］,［AVコントロール］

モード　条0≦モード<4。数値型定数，変数，配列変数，式。
　　　　省0

輝度　条0≦輝度<2。数値型定数，変数，配列変数，式。
　　　　省0

カラーバス　条0≦カラーバス<2。数値型定数，変数，配列変数，式。
　　　　省0

同期　条0≦同期<2。数値型定数，変数，配列変数，式。
　　　　省0

音声　条0≦音声<4。数値型定数，変数，配列変数，式。
　　　　省0

ビデオ入力　条0≦入力<2。数値型定数，変数，配列変数，式。
　　　　省0

AVコントロール　条0≦コントロール<2。数値型定数，変数，配列変数，式。
　　　　省1

S

働きと使いかた

モードと働きは次のようになっています。

モード	S1	S0	TP	同期	機能
0	0	0	0	内部	コンピューター
1	0	1	1	外部	コンピューター
2	0	1	0	外部	スーパーインポーズ
3	1	0	0	外部	テレビ

●S1，S0，TPはVDPレジスターフラグの名前です。

モードが0のとき外部同期をかけることはできません。また，1～3のときはVDPのコンポジット出力は使用できません。アナログRGB出力を使用してください。

輝度　0のとき通常の輝度になります。

　　　1のときテレビの輝度は半分になります。

カラーバス　0のときVDPのカラーバスが出力状態になります。

　　　　　　1のときVDPのカラーバスが入力状態になります。

同期　0のとき内部同期。

　　　1のとき外部同期。

S

音声

音声	機能
0	外部音声信号を混合しない。
1	右チャンネルの外部音声信号を混合する。
2	左チャンネルの外部音声信号を混合する。
3	両チャンネルの外部音声信号を混合する。

ビデオ入力　0のときRGBマルチコネクターの入力を選択。
　　　　　　1のとき映像入力端子の入力を選択。

AVコントロールはRGBマルチコネクターのAVコントロール端子の出力を指定します。0のときテレビを指定し，1のときコンピューターを指定します。

S

関数 SGN (sign)

(エスジーエヌ)

数値データが正のとき1，0のとき0，負の時−1を返す。

形式

SGN(X)

X 条数値型定数，変数，配列変数，式。

返される値：整数型。

実行例

```
10 INPUT A
20 IF SGN(A)=-1 THEN PRINT "Negative"
30 GOTO 10
```

行番号20で，Aに代入された値が負数のときだけ″Negative″と表示させています。

関数 SIN (sine)

(サイン)

数値データの正弦（サイン）の値を返す。

関TAN，COS，ATN

形式

SIN(X)

X 条数値型定数，変数，配列変数，式。(単位ラジアン)

返される値：−1≦値≦1。浮動小数点型定数。

実行例

```
PRINT SIN(3.14/3)
 .86575983949239
PRINT SIN(60*3.14/180)
 .86575983949239
```

●Xを度（degree）の単位で与えるときはSIN（X＊円周率/180）とします。

S

SOUND（sound）

サウンド

PSG（音出しの専門のLSI）のレジスターに直接データを書き込んで効果音などを出す。

関BEEP，PLAY

形　式

SOUND レジスター番号，式

レジスター番号　条0≦番号＜14。数値型定数，変数，配列変数，式。

式　条各レジスターごとに定められた範囲の数値型定数，変数，配列変数，式。

働きと使いかた

PSGの各レジスターの働きと書き込めるデータの範囲

レジスター番号	働　き	データの範囲
0	チャンネルAの周波数	0-255
1		0-15
2	チャンネルBの周波数	0-255
3		0-15
4	チャンネルCの周波数	0-255
5		0-15
6	ノイズの周波数	0-31
7	どのチャンネルからトーンやノイズを出すかを選択	0-63
8	チャンネルAの音量	0-15 16にすると 音量変化
9	チャンネルBの音量	
10	チャンネルCの音量	
11	音量変化パターンの周波数	0-255
12		0-255
13	音量変化のパターン選択	0-14

S

周波数を決めるには

0から5までの6つのレジスターを使って，3つのチャンネルから出す音の周波数を決めますが，レジスターに書き込むべきデータは次の式から得ることができます。

$$\frac{1789772.5\ (\mathrm{Hz})}{16\times \text{出力したい周波数}\ (\mathrm{Hz})}=256\times(\text{レジスター1, 3, 5のデータ})$$

$$+(\text{レジスター0, 2, 4のデータ})$$

たとえば，チャンネルAから300Hzの音を出したければ

$$\frac{1789772.5\ (\mathrm{Hz})}{16\times 300}\fallingdotseq 373=256\times 1+117$$

という式が成り立ちますから，レジスター0に117，レジスター1に1を書き込めば良いことになります。

ノイズの周波数を決めるには

レジスター6には0から31までのデータを書き込むことができます。このデータにより，ノイズ（ザーッという音）の周波数が決まりますが，データと周波数の間には次の関係が成り立ちます。

$$\text{データの値}=\frac{1789772.5\ (\mathrm{Hz})}{16\times \text{ノイズの周波数}\ (\mathrm{Hz})}$$

たとえば，レジスター6にデータ15を書き込むと

$$15\fallingdotseq\frac{1789772.5\ (\mathrm{Hz})}{16\times 7457}$$

ですからノイズ周波数は約7457Hzになります。

どのチャンネルを使うかを指定するには

レジスター7に書き込むデータによって，どのチャンネルを使うかを決めます。

ノイズ			音		
チャンネル C	B	A	C	B	A
32	16	8	4	2	1

上の表から，使うチャンネルに対応する数値を合計して，63から引くと書き込むべきデータが求められます。

たとえば，チャンネルA，Bからは音だけ，チャンネルCからは音とノイズを出したければ，

63－（32＋4＋2＋1）＝24

で，24が書き込むデータとなります。

S

音量を決めて音を出す

レジスター8，9，10には，それぞれチャンネルA，B，Cの音量を決めるデータを書き込みます。0から15までのデータが書き込め，15のときに最大の音量となります。

音量変化パターンにより，効果音を作るには

PLAY文のSサブコマンドとMサブコマンドと同じことがSOUND文でもできます。まず，音量変化パターンはレジスター13に書き込むデータで決りますが，これはPLAY文のSサブコマンド(Sn)におけるnの指定と同じです。パターンに対応するnの値は133ページの表を見てください。

次に，音量変化パターンの周期ですが，これはレジスター11,12に書き込むデータで決まり，下の式が成りちます。

$$\frac{1789772.5\ (\text{Hz})}{256\times\text{周期}\ (\text{Hz})}=256\times\ (\text{レジスター12のデータ})+\ (\text{レジスター11のデータ})$$

例えば，周期を10Hzにしたい場合は

$$\frac{1789772.5}{256\times10}\fallingdotseq699=256\times2+187$$

ですから，レジスター11に187，レジスター12に2を書き込めば良いことになります。

実行例

```
10 SOUND 0,56   } チャンネルAの周波数を400Hzに
20 SOUND 1,1
30 SOUND 7,62   ──チャンネルAのトーンを選択
40 SOUND 8,8    ──チャンネルAの音量を指定
```

このプログラムを実行すると400Hzの音が出力され続けるようになります。止めるには CTRL + STOP を実行します。

S

関数 SPACE$ (Space dollar)

スペースダラー

任意の数のスペースを文字データとして返す。　　関SPC

形　式

SPACE$(N)

N　条0≦N<256。数値型定数，変数，配列変数，式。

返される値：文字型

実行例

```
PRINT SPACE$(5);"ABC"
     ABC
```

5個のスペース

●Nが整数値でない場合は，小数点以下が切り捨てられます。

関数 SPC(space)

スペース

任意の数のスペースを出力する。　　関SPACE$

形　式

SPC (N)

N　条0≦N<256。数値型定数，変数，配列変数，式。

返される値：文字型。

働きと使いかた

SPC関数はPRINT文とLPRINT文でのみ使用可能です。

実行例

```
PRINT "ABC";SPC(10);"DEF"
ABC          DEF
```

10個のスペース

●Nが整数値でない場合は，小数点以下が切り捨てられます。

S

SPRITE ON (sprite on)
SPRITE OFF (sprite off)
SPRITE STOP (sprite stop)

(スプライト オン / スプライト オフ / スプライト ストップ)

スプライトの衝突による割り込みを有効，無効，または保留にする。

関PUT SPRITE

形　式

SPRITE ON――――――割り込み有効

SPRITE OFF――――――割り込み無効

SPRITE STOP――――――割り込み保留

働きと使いかた

ON SPRITE GOSUB文でスプライトの衝突による割り込みを宣言したあと，実際に割り込みを有効に（SPRITE ON），あるいは無効（SPRITE OFF），保留（SPRITE STOP）にするためのコマンドです。

関　数

SQR (square root)

(スクエア)

数値データの平方根の値を返す。

形　式

SQR(X)

X　条0≦X。数値型定数，変数，配列変数，式。

返される値：数値型。

実行例

```
PRINT SQR(100)
 10
```

S

関 数 # STICK (stick)

スティック

カーソルキー，ジョイスティックの方向を返す。 関STRIG

形 式

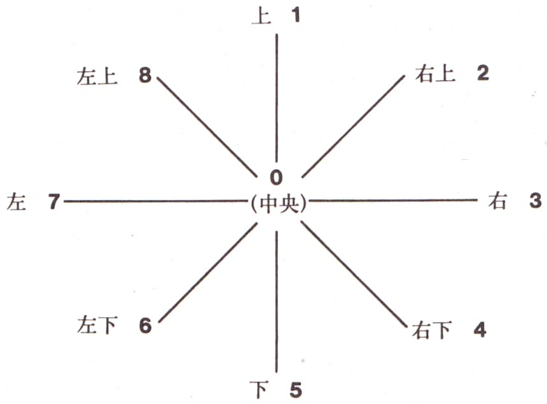
STICK(N)

N 条0≦N<3。数値型定数，変数，配列変数，式。

返される値：整数型

働きと使いかた

N=0のときはカーソルキー， N=1のときはジョイスティックA， N=2のときはジョイスティックBの方向を返します。返される値は0から8までで，その値により方向を知ることができます。どのカーソルキーも押されていないとき，あるいはジョイスティックが中央の位置にあるときは0が返されます。



実行例

```
10 CLS
20 X=14
30 LOCATE X,10:PRINT " ";
40 D=STICK(0)
50 IF D=0 THEN LOCATE X,10:PRINT "*"
60 IF D=3 THEN X=X+1:IF X>28 THEN X=28
70 IF D=7 THEN X=X-1:IF X<0 THEN X=0
80 LOCATE X,10:PRINT "*";
90 GOTO 30
```

左右のカーソルキーを使って，画面上の〝*〞を左右に動かすプログラムです。カーソルキーが押されているかどうかにより行番号40で変数Dに返される値が異なります。行番号50, 60, 70で，返された値により〝*〞を表示させる位置のX座標を変更しています。

S

STOP（stop）

プログラムの実行を中断する。

形　式

STOP

働きと使いかた

プログラムがSTOP文にぶつかると，そこで実行が中断します。

●直接モードでCONT文を実行すると中断した次の行から実行を再開します。

STOP ON（stop on）
STOP OFF（stop off）
STOP STOP（stop stop）

CTRL＋STOPキーによる割り込みを有効，無効または保留にする。　関END，CONT

形　式

STOP ON――――割り込み有効

STOP OFF――――割り込み無効

STOP STOP――――割り込み保留

働きと使いかた

ON STOP GOSUB文で，CTRL＋STOPキーによる割り込みを宣言したあと，実際に割り込みを有効に（STOP ON），あるいは無効（STOP OFF），保留（STOP STOP）にするためのコマンドです。

S

関数 STRIG（stick trigger）

エストリッグ

コントローラー（ジョイスティック，マウス，トラックボール）のボタンが押されていれば−1を，押されていなければ0を返す。　関STICK

形式

STRIG(N)

N　条0≦N＜5。数値型定数，変数，配列変数，式。

返される値：整数型

働きと使いかた

N＝0のときはスペースバー，N＝1，N＝3　のときはコントローラーAのボタン，N＝2，N＝4のときはコントローラーBのボタンの状態を返します。返される値は0または−1で，押されていないときが0，押されているときが−1です。

実行例

```
10 CLS
20 COLOR ,C,C
30 IF STRIG(0)=0 THEN GOTO 20
40 C=C+1:IF C>15 THEN C=0
50 GOTO 20
```

スペースバーを押すたびに画面の色が変わるプログラムです。

STRIG ON (stick trigger on)
STRIG OFF (stick trigger off)
STRIG STOP (stick trigger stop)

スペースバー，コントローラー（ジョイスティック，マウス，トラックボール）のボタンによる割り込みを有効，無効または保留にする。 関STRIG

形　式

STRIG (N) ON――――割り込み有効

STRIG (N) OFF――――割り込み無効

STRIG (N) STOP――――割り込み保留

N　条0≦N<5。数値型定数，変数，配列変数，式。

働きと使いかた

Nで，スペースバー，コントローラーA，Bのボタンのどれを割り込みに使うかを指定します。このとき，ON STRIG GOSUB文により，対応するサブルーチンの行番号が指定されていなければなりません。

nの値	指定されるもの
0	スペースバー
1	コントローラーAのボタン1
2	コントローラーBのボタン1
3	コントローラーAのボタン2
4	コントローラーBのボタン2

```
STRIG(0) ON
STRIG(1) OFF
STRIG(2) STOP
```

STRIG(0) ON…………スペースバーによる割り込みを有効に

STRIG(1) OFF………コントローラーAのボタン1による割り込みを無効に

STRIG(2) STOP……コントローラーBのボタン1による割り込みを保留に

関　数　STR$（convert to string）

エスティーアールダラー

数値データを文字データに変換して返す。　反VAL

形　式

STR$(X)

X　条数値型定数，変数，配列変数，式。

返される値：文字型。

働きと使いかた

数値データが負数の場合，返される文字データの1字目は－になります。0または正数の場合，返される文字データの1字目はスペースになります。

実行例

```
10 X=100:Y=200
20 X$=STR$(X):Y$=STR$(Y)
30 PRINT X+Y
40 PRINT X$+Y$
RUN
 300
 100  200
 X$   Y$
```

S

関　数　STRING$ (string dollar)

ストリングダラー

与えられた文字コードの文字，または与えられた文字列の先頭の文字を，任意の数だけ続けて，文字列データとして返す。 関CHR$

形　式

STRING$ (N, J)

STRING$ (N, X$)

N　　条0≦N<256。数値型定数，変数，配列変数，式。

J　　条任意の文字コード（258ページの文字コード表参照）

X$　　条文字型定数，変数，配列変数，式。

返される値：文字型。

実行例

```
PRINT STRING$(10,70)
FFFFFFFFFF
PRINT STRING$(5,"ABC")
AAAAA
```

SWAP (swap)

スワップ

2つの変数の値を交換する。

形　式

SWAP 変数，変数

変数　条数値型または文字型の変数，配列変数。2つの変数の型は同じであること。

実行例

```
10 A=3:B=5
20 SWAP A,B
30 PRINT "A=";A
40 PRINT "B=";B
RUN
A= 5
B= 3
```

関　数　TAB（tab）

（タブ）

カーソルを指定した文字数だけ行頭から右へ移動する。

形　式

TAB(N)

N　条0≦N＜256。数値型定数，変数，配列変数，式。

働きと使いかた

TAB関数はPRINT文とLPRINT文でのみ使用可能です。

Nが0のとき左端，1行の文字数より1少ない値が右端となります。

実行例

```
PRINT TAB(5);"AAA"
     AAA
```

5字分

関　数　TAN（tangent）

（タンジェント）

数値データの正接（タンジェント）の値を返す。　反ATN　関COS，SIN

形　式

TAN(X)

X　条数値型定数，変数，配列変数，式。（単位ラジアン）

返される値：浮動小数点型定数。

実行例

```
PRINT TAN(3.14/3)
 1.72992922009
PRINT TAN(60*3.14/180)
 1.72992922009
```

●Xを度（degree）の単位で与えるときはTAN（X＊円周率/180）とします。

T

TROFF (trace off)

(トレースオフ)

TRONを解除し，実行した行番号の表示をやめる。　反TRON

形　式

TROFF

働きと使いかた

すでにTRON文が実行されているときに，直接モードあるいは間接モードでTROFF文を実行すると行番号の表示が解除されます。

TRON (trace on)

(トレースオン)

実行した行番号を表示する。　反TROFF

形　式

TRON

働きと使いかた

直接モード，あるいは間接モードで，TRON文が1回実行されると，以後実行された行番号がテキストモードの画面上に[　]に囲まれて表示されるようになります。プログラムのデバック（間違いの修正）をするときなどに使います。

● 画面がSCREEN文によりグラフィックモードになっているときには行番号は表示されません。

実行例

```
10 TRON
20 FOR I=0 TO 3
30 A=I+1:PRINT A
40 NEXT I
50 TROFF
RUN
[20][30] 1
[40][30] 2
[40][30] 3
[40][30] 4
[40][50]
```

T

関　数　USR（user）

ユーザー

DEFUSR文で定義した番地から始まる機械語サブルーチンを実行し，得られた結果を返す。

関DEFUSR

形　式

USR［X］（I）

X　　条0≦X≦9。整数型定数。

　　　省0

I　　条数値型または文字型の定数，変数，配列変数。

返される値：ユーザー定義関数による。

働きと使いかた

XはユーザープログラムBASIC番号です。DEFUSR文で指定した番号を使います。IはBASICからサブルーチンに引き渡したい値を表す変数または定数です。

機械語サブルーチンの実行

変数＝USRN（I）

を実行することにより，定義した機械語サブルーチンが実行されます。機械語サブルーチンの実行が終了すると，実行結果の値が変数に返されるとともに，引き続きBASICプログラムが実行されます。

機械語サブルーチンに実行が移されるときに，USR関数のパラメーターとして指定した“I”の値がサブルーチンに引き渡されます。

X＝USR 1（I）

とすると，変数Iの値はメモリー上の次のような位置に格納されるとともに，Iの型によってレジスターに型を示すデータが入ります。HLレジスターには，Iの値が格納される領域の先頭アドレスが入ります。

U

Iの型	Aレジスターに入るデータ*	HLレジスターの示すアドレス	Iの値が格納されるアドレス
整数型	2	&HF7F6	&HF7F8-&HF7F9
単精度型	4		&HF7F6-&HF7F9
倍精度型	8		&HF7F6-&HF7FD

*同じデータがメモリーの&HF663番地にも入ります。

Iが文字型変数の場合は次のようになります。

Aレジスターに入るデータ	DEレジスターに入るデータ	ストリングデスクリプター
3	ストリングデスクリプターの先頭のアドレス	1バイト目：文字列の長さ 2,3バイト目：文字列の格納された領域の先頭アドレス

機械語サブルーチンの実行が終了すると，終了時にレジスター，メモリーを次のようにしておくことにより，結果の値は変数Xに返されることになります。

結果の値の型	メモリーの&HF663番地	DEレジスター	HLレジスター	結果を格納するアドレス
整数型	2		&HF7F6	&HF7F8-&HF7F9
単精度型	4		&HF7F6	&HF7F6-&HF7F9
倍精度型	8		&HF7F6	&HF7F6-&HF7FD
文字型	3	ストリングデスクリプターの先頭アドレス		ストリングデスクリプターの2，3バイト目で示されるアドレスを先頭とする領域

実行例

```
DEFUSR0=&HE000
```

としてから

```
X=USR0(I)
```

すると，&HE000番地以降のサブルーチンが実行され，その結果の値がBASICに返されます。

関　数 VAL (value)

バリュー

文字データを数値データとして返す。 関STR$

形　式

VAL(X$)

X$　条数字を表す文字型定数，変数，配列変数，式。

返される値：数値型。

実行例

```
PRINT VAL("5")
 5
PRINT VAL("       5")
 5
```

文字型データの前のスペースは無視される。

関数 VARPTR（variable pointer）その1

バリアブルポインター

指定した変数に代入されているデータが格納されているメモリーの先頭番地を返す。

形式

VARPTR（変数）

変数　条数値型，文字型の変数，配列変数。

返される値：－32768≦変数≦32767　整数型定数。

働きと使いかた

変数に代入されている値が格納されているメモリーの先頭番地を10進数で返します。返された値が負数の場合は，その値に65536を加えたものが実際の番地になります。この関数は，機械語サブルーチンにデータの入っているメモリーの番地を引き渡すときなどに使います。

実行例

```
10 A%=15
20 X=VARPTR(A%)
30 M$=HEX$(X):PRINT M$
RUN
802E
```

変数A%に代入した値15が格納されているメモリーの番地を調べ，それを16進形式に変換して表示させるプログラムです。

VARPTRを使う前にプログラムで調べたい変数に値を代入しておく必要があります。（変数に値を代入した時点で変数の番地が変化します）

関数 VARPTR（variable pointer）その2

（バリアブルポインター）

指定したファイルが割り当てられているファイルコントロールブロックの先頭番地を返す。

形　式

VARPTR（#ファイル番号）

ファイル番号　条1≦ファイル番号≦MAXFILES＝文で指定された数。整数型定数，変数，配列変数，式。

返される値：−32768≦値≦−1。整数。

働きと使いかた

指定したファイルが割り当てられているファイルコントロールブロックの先頭番地を返します。

ファイルコントロールブロックには，そのファイルへの入出力用に設定された領域（ファイルバッファ）の先頭番地が格納されています。

関数 VPEEK（video RAM peek）

ブイピーク

ビデオRAMの中のデータを読み取る。

形式

VPEEK（アドレス）

アドレス　条0≦アドレス＜65536。数値型定数，変数，配列変数，式。

働きと使いかた

ビデオRAMの指定されたアドレスに書き込まれているデータを返します。
SCREEN 5から8およびSCREEN 10から12のときには，アドレスにアクティブページの先頭アドレスを加えた値がビデオRAMの絶対アドレスになります。

スクリーンモード	アクティブページ	先頭アドレス
5，6	0	0
	1	&H08000
	2	&H10000
	3	&H18000
7，8	0	0
	1	&H10000

実行例

SCREEN 6の&H10200番地の内容を知りたい時には以下のようにします。

```
10 SCREEN 6
20 SET PAGE ,2
30 A=VPEEK(&H200)
40 SCREEN 0
50 PRINT A
```

行20でアクティブページを2ページにセットします。
行30でビデオRAMの内容を読んでいますが，ビデオRAMの絶対アドレスは，&H200（vpeekのアドレス）＋&H10000（アクティブページの先頭アドレス）＝&H10200となります。

ブイポーク

VPOKE（video RAM poke）

ビデオRAMに1バイトのデータを書き込む。

形　式

VPOKE アドレス，式

アドレス　条0≦アドレス＜65536。数値型定数，変数，配列変数，式。

式　条0≦式の値＜256。数値型定数，変数，配列変数，式。

働きと使いかた

ビデオRAMの指定したアドレスに任意のデータを書き込みます。

SCREEN 5から8およびSCREEN 10から12のときは，アドレスにアクティブページの開始アドレスを加えた値がビデオRAMの絶対アドレスになります。

ビデオRAMのアドレスマッピングについては，BASE関数により各テーブルのベースアドレスを知ることができますので，VPOKE文を使用するときは目的のビデオRAMのアドレスをBASE関数で調べてください。

WAIT (wait)

ウエイト

I/Oポートからの入力が，ある値になるまで待つ。

形　式

WAITポート番号，式1［，式2］

ポート番号，式1，式2　条0≦番号／式<256。数値型定数，変数，配列変数，式。

働きと使いかた

WAIT文が実行されると，指定したI/Oポートからデータを入力して，式2の値とのXOR（排他的論理和）をとり，次にその結果と式1の値とのAND（論理積）をとります。こうして得られた値が0なら，ひき続きI/Oポートからのデータ入力を続け，0以外になれば次の行番号に進みます。

式2を省略すると，式2の値は0とみなされます。

WIDTH (width)

ウィドス

テキストモードでの一行の文字数を指定する。

形　式

WIDTH　字数

字数　条SCREEN0のとき

1≦数<81　数値型定数，変数，配列変数，式。

SCREEN 1のとき

1≦数<32　数値型定数，変数，配列変数，式。

働きと使いかた

SCREEN 0のとき指定された文字数が40以下ならば40字×24行テキストモードが選択され，41以上ならば80字×24行テキストモードが選ばれます。

実行例

```
SCREEN 0
WIDTH 40
```

ご注意

日本語入力モードにおけるWIDTHについては240ページを参照してください。

特別な変数と関数

BASE（base）

（ベース）

VDPテーブルのベースアドレスの読み出し，書き込みに使用する。

形　式

BASE（N）

N　条値を読むとき0≦N≦44，50≦N≦64。数値型定数，変数，配列変数，式。
　　値を書き込むときに0≦N≦19。数値型定数，変数，配列変数，式。

働きと使いかた

値を書き込むときはSCREENモード3までです。

Nは次のように計算します。

　SCREENモード＊5＋テーブル番号

例えばSCREEN 2のスプライト属性テーブルの（テーブル番号3）アドレスを知るためには次のようにNの値を出します。

　N＝2＊5＋3
　　＝13

テーブル番号表

テーブル番号	テーブルの種類
0	パターン名称テーブル
1	カラーテーブル
2	パターンジェネレーターテーブル
3	スプライト属性テーブル
4	スプライトパターンテーブル

Nの値がわかっていてSCREENモードやアドレスを知るためには次の式を使います。

　SCREENモード　INT（N／5）
　テーブル番号　（N MOD 5）

例えばN＝44のとき
　44／5＝8 余り4

したがってN＝44のときのSCREENモードは8，テーブル番号は4となります。

SCREEN 2モードのテーブルアドレスを変更すると，同時にSCREEN 4モードのテーブルアドレスも変化します。

SCREEN 5～8，10～12モードの場合，読み出されたアドレスはアクティブページの先頭アドレスからのオフセット値です。

MAXFILES (maximum files)

マックスファイルズ

ひとつのプログラム中で同時に開くことのできるファイルの数を宣言する。 関OPEN

形 式

MAXFILES＝式

式 条0≦式＜16 数値型定数，変数，配列変数，式。

働きと使いかた

ひとつのプログラムで同時に開くことのできるファイルの数を宣言します。同時にファイルを開くとは，ひとつのファイルをOPENし，それをCLOSEする前に他のファイルをOPENすることをいいます。MAXFILES＝文は必ずプログラムの先頭で宣言してください。

実行例

```
10 MAXFILES=3
20 OPEN "GRP:" FOR OUTPUT AS #1
30 OPEN "CRT:" FOR OUTPUT AS #2
40 OPEN "LPT:" FOR OUTPUT AS #3
              .
              .
              .
1000 CLOSE
```

行番号10で，開くことのできるファイルの数を3にしたため，行番号20以下で3つのファイルを開くことができます。

MAXFILES＝文でファイルの数を指定しないときは，同時にひとつのファイルしか開くことはできません。

●不必要に大きな値を宣言すると，それだけユーザーエリアが少なくなってしまいます。

SPRITE$（sprite dollar）

スプライトダラー

スプライトパターンのデータを定義する。　　関PUT SPRITE

形　式

SPRITE$（スプライト番号）

スプライト番号　条8×8ドットのとき…0≦番号＜256。数値型定数，変数，配列変数，式。
16×16ドットのとき…0≦番号＜64。数値型定数，変数，配列変数，式。

働きと使いかた

1度SPRITE$変数にスプライトパターンを定義しておくと，指定したスプライト番号のパターンとして保持されます。定義のしかたについてはPUT SPRITEの項をご覧ください。

TIME (time)
タイム

内蔵タイマーの値を保持する。　関SET TIME

形　式

TIME

TIME ＝式

式　条0≦式＜65536。数値型定数，変数，配列変数，式。

特別
T

働きと使いかた

この変数はBASIC起動時にすでに内蔵タイマーの値を保持しています。タイマーの値は0以上65536未満で，約1/60秒ごとに1ずつ繰り上がっていきます。65535になると，次は再び0になります。

この変数の値はLET文により書き変えることができます。なお，このタイマーはCPUが割り込み禁止状態にあるとき（カセットテープへの入出力中など）は停止しています。また電源を切ると働きません。

実行例

```
10 CLS:TIME=0
20 LOCATE 12,8:PRINT INT(TIME/60)
30 GOTO 20
```

TIME変数の値をいったん0にしてから，60で割った値の整数部分を次々に表示していくプログラムです。約1秒ごとに1ずつ数が繰り上がっていきます。

VDP (video display processor)

ブイディピー

VDPレジスターの内容を読み出したり書き込んだりする。

形　式

VDP (N)

N　条0≦N≦28, 33≦N≦47, −9≦N≦−1　整数型定数。

働きと使いかた

MSXパーソナルコンピューターの画像表示用LSI(VDP)のレジスター内容を読み出したり, 設定したりします。

N	レジスター	機能
0〜7	0〜7	設定可
8	ステータスレジスター　0	読み込み専用
9〜28	8〜27	設定可
33〜47	32〜46	設定可
−1〜−9	ステータスレジスター　1〜9	読み込み専用

ご注意

VDP変数を使ってVDPレジスターの値を書き換えて画面の操作をするには, VDPに関する充分な知識が必要です。不用意にVDPレジスターの値を書き換えると, 画面表示が正しく行われなくなりますのでご注意ください。

特別
V

フロッピーディスクを使うための拡張コマンド

フロッピーディスクを使う前には必ずフォーマットをしなくてはなりません。MSX-Disk BASICには，このような作業を実行するためのFORMATコマンドと，オペレーティングシステムへ制御を渡すためのSYSTEMコマンドが拡張コマンドとして準備されています。
拡張コマンドは，いずれもCALL文により実行することができます。

FORMAT（format）

ディスクをフォーマットする。

形　式

CALL FORMATまたは

_FORMAT

働きと使いかた

ディスクに，定められたデータを一定のルールに従って書き込むことをフォーマットするといいます。MSX-Disk BASICは，フォーマットによって書き込まれたデータを目印にしてファイルの名前やファイルの内容を探し出したり書き込んだりします。したがって，新しいディスクを使うときは，必ず最初にFORMATコマンドでフォーマットしておかなくてはなりません。

```
CALL FORMAT
```

または

```
_ FORMAT
```

を実行すると，

```
Drive name? (A,B)
```

という表示があらわれます。
ここで，フォーマットしたいディスクの入っているドライブ名（AまたはB）をキーボードから指定します。
指定すると，片面用のディスクドライブがつながれているときは次の表示があらわれます。

```
Strike a key when ready
```

フォーマットしたいディスクが，指定したドライブに入っていることを確かめ，（入っていなかったら入れて）キーボード上のキーを何かひとつ押します。
両面用のディスクドライブがつながれているときは，

```
1 - Single sided, 9 sectors
2 - Double sided, 9 sectors
```

という表示があらわれますので，片面のディスクをフォーマットするときは1を，両面のディスクをフォーマットするときは2を指定してください。
指定すると，次の表示があらわれます。

```
Strike a key when ready
```

片面用ディスクドライブがつながれたときと同様にキーを何かひとつ押してください。
ディスクがフォーマットされ，終了すると，

```
Format complete
Ok
```

と表示され，コマンド待ち状態に戻ります。

● ディスクのフォーマットを途中で止めるとき（Strike a key when readyの表示以前の時）は[CTRL]キーと[STOP]キーを同時に押します。

ディスクドライブが1台の場合

ドライブ名を指定するときはAを指定します。ただし，Bを指定すると，1台のドライブを仮想的に2台のドライブとみなし，

```
Insert diskette for drive B:
and strike a key when ready
```

と表示されます。この時点で，1台のドライブがBドライブとみなされたことになります。
以下，このドライブを使ってディスクをフォーマットすることができます。

SYSTEM（system）

システム

MSX-DOSに制御を移す。

形　式

CALL SYSTEMまたは
_SYSTEM

働きと使いかた

すべてのファイルを閉じ，メモリーの中のプログラムやデータをクリアしてからMSX-DOSに制御を移します。
ディスクドライブにMSX-DOSのシステムファイル（MSXDOS.SYSとCOMMAND.COM）の入ったディスクがセットされ，MSX-DOSからBASICが起動された場合だけMSX-DOSに戻ることができます。

拡張

メモリーディスクを使うための拡張コマンド

メモリーディスクを初期化し，ファイルを管理するためにはBASICの拡張が必要です。拡張コマンドとして次のようなコマンドが準備されています。

MEMINI，MFILES，MKILL，MNAME

メモリーディスクのファイルをアクセスするには，デバイス名をMEM：として通常のコマンドを使います。
拡張コマンドは，いずれもCALL文により実行することができます。

MEMINI（memory disk initialize）

（メムイニ）

メモリーディスクが使うメモリーを割り当て，メモリーディスクを初期化し，メモリーディスク内の全ファイルを消去する。

形　式

CALL MEMINI［(サイズ)］
または
_MEMINI［(サイズ)］

サイズ　条1023≦サイズ＜32768。数値型定数，変数，配列変数，式。
省32767

働きと使いかた

RAMの0000H番地から7FFFH番地までをメモリーディスクとして使えます。メモリーディスクの機能を使うためには事前にこの命令を実行する必要があります。
サイズはメモリーディスクが使うメモリー容量をバイト単位で指定します。
CALL MEMINI（0）を実行するとメモリーディスクの機能が解除され，メモリーディスクは使えなくなります。

実行例

CALL MEMINI または _MEMINI を実行すると

```
32000 bytes allocated
```

と表示される。

拡張

MFILES (memory disk files)

(エムファイルズ)

メモリーディスクに記憶されているファイル名を表示する。

形　式

CALL MFILES

または

_MFILES

MKILL (memory disk kill)

(エムキル)

メモリーディスク内の指定したファイルを消す。

形　式

CALL MKILL ("ファイル名 [.タイプ名]")

または

_MKILL

ファイル名　条8字までの文字列。

タイプ名　条3字以内の文字列。

省ヌルストリング。

実行例

```
CALL MKILL ("TEST.BAS")
```

または

```
_MKILL ("TEST.BAS")
```

MNAME (memory disk name)

エムネーム

メモリーディスク内のファイル名を変更する。

形　式

CALL MNAME ("旧ファイル名 [.旧タイプ名]" AS "新ファイル名 [.新タイプ名]")

または

_MNAME ("旧ファイル名 [.旧タイプ名]" AS "新ファイル名 [.新タイプ名]")

旧ファイル名，新ファイル名　条8字までの文字列。
旧タイプ名，新タイプ名　条3字以内の文字列。
省ヌルストリング。

実行例

```
CALL MNAME ("TEST.BAS" AS "SONY.BAS")
```

または

```
_MNAME ("TEST.BAS" AS "SONY.BAS")
```

拡張

FM音源に関する拡張コマンド

FM音源(MSX-MUSIC)を内蔵してあるか，取り付けてあるMSXコンピューターで，FM音源を使った音楽演奏を行うために，次の拡張コマンドが準備されています。
拡張コマンドは，いずれもCALL文により実行することができます。

♪ AUDREG(audio register)

オーディオレジスター

FM音源用LSIのレジスターに直接データを書き込む。

形　式

CALL AUDREG(レジスター番号,式［,チャンネル番号］)　または

_AUDREG(レジスター番号,式［,チャンネル番号］)

レジスター番号　条0～7，14，15，16～24，32～40，48～56の数値型定数，変数，配列変数，式。

式　条各レジスターごとに定められた範囲の数値型定数，変数，配列変数，式。

チャンネル番号　条必ず0にする。(あるいは省略する。)

働きと使いかた

FM音源用LSIのレジスターに，データを書き込んで操作します。
このコマンドを使用するためには，MSX-MUSICのFM音源LSIの仕様に関する知識が必要です。

拡張 ♪

♪ BGM(bgm)

(ビージーエム)

バックグラウンドで演奏するかフォアグラウンドで演奏するかを指定する。

形　式

CALL BGM(N) または

_BGM(N)

N　条0または1。数値型定数，変数，配列変数，式。

働きと使いかた

Nの値を0にすると，PLAY文での演奏はフォアグラウンドで行われます。この場合，演奏中はプログラムの実行は停止し，演奏が終了すると次の行に実行が進みます。
Nの値を1にすると，PLAY文での演奏はバックグラウンドで行われます。この場合，演奏が始まるとともにプログラムは次の行に進み，音楽の演奏とともに他の処理が進行します。

♪ MUSIC(music)

(ミュージック)

FM音源を使用可能にする。

形　式

CALL MUSIC[([モード][,[0][,PLAY文第1サブコマンド列用のチャンネル数[,第2サブコマンド列用のチャンネル数[,...,第9サブコマンド列用のチャンネル数]]]]]]]]]])]　または

_MUSIC[([モード][,[0][,PLAY文第1サブコマンド列用のチャンネル数[,第2サブコマンド列用のチャンネル数[,...,第9サブコマンド列用のチャンネル数]]]]]]]]]])]

モード　　条0または1。数値型定数，変数，配列変数，式。
　　　　　省すべてのパラメーターを省略したときは1。
　　　　　モード以外のパラメーターを指定し，モードは省略したときは0。

チャンネル数　省モードが0のとき:
　　　　　1≦チャンネル数≦9。数値型定数，変数，配列変数，式。
　　　　　モードが1のとき:
　　　　　1≦チャンネル数≦6。数値型定数，変数，配列変数，式。

働きと使いかた

CALL MUSIC文を実行することにより，FM音源用LSIが初期化されます。CALL MUSIC文により初期化を行なうまで，FM音源演奏用の拡張コマンドおよびFM音源に関するPLAY文は実行できません。

モードについて

モードを0にするとリズム音を使用しないモード，1にするとリズム音を使用するモードになります。リズム音とはドラムやシンバルなどのリズム楽器を模した音で，PLAY文中でリズム音のサブコマンドとして指定します。

チャンネル数の指定のしかた

FM音源にはチャンネル1からチャンネル9までの9チャンネルがあります。リズム音を使用する場合は，リズム音がチャンネル7からチャンネル9までの3チャンネルから同時に出力されます。したがって，リズム音を使用しない場合はPLAY文のサブコマンドによるメロディーをチャンネル1から9まで，リズム音を使用する場合はメロディーをチャンネル1から6までを使って出力することができます。

CALL MUSIC文で指定する数だけ，PLAY文のサブコマンド列にチャンネルが順次割り当てられます。PLAY文のサブコマンド列で，FM音源のチャンネルに割り当てられないものは，PSGから出力されます。

例1

リズム音を使用しないモードで，

**PLAY #2, A$, B$, C$, D$, E$**（A$，B$，C$，D$，E$はサブコマンド列を表わす）

という**PLAY**文を実行する場合，**CALL MUSIC**文によるチャンネル数指定と，各サブコマンド列のメロディーが出力される**FM**音源のチャンネルおよび**PSG**との関係は以下のようになります。

CALL MUSIC文	サブコマンド列	出力チャンネル
CALL MUSIC（0,0,1,1,1,1,1）	A$	チャンネル1
	B$	チャンネル2
	C$	チャンネル3
	D$	チャンネル4
	E$	チャンネル5
CALL MUSIC（0,0,3,2,1,2,1）	A$	チャンネル1，2，3
	B$	チャンネル4，5
	C$	チャンネル6
	D$	チャンネル7，8
	E$	チャンネル9
CALL MUSIC（0,0,4,5）	A$	チャンネル1，2，3，4
	B$	チャンネル5，6，7，8，9
	C$，D$，E$	PSG音源の3つのチャンネルからそれぞれ出力される

例2

リズム音を使用するモードでのCALL MUSIC文によるチャンネル指定とPLAY文のサブコマンド列との対応例を以下に示します。実行するPLAY文は

PLAY #2, A$, B$, C$, D$ (A$, B$, C$はメロディーのサブコマンド列, D$はリズム音のサブコマンド列)

とします。

CALL MUSIC文	サブコマンド	出力チャンネル
CALL MUSIC (1,0,2,2,2)	A$	チャンネル1, 2
	B$	チャンネル3, 4
	C$	チャンネル5, 6
	D$	チャンネル7, 8, 9

なお, チャンネル数の指定を0にしたり, 途中を省略することはできません。

CALL MUSIC(0,0,1,0,3)のようにチャンネル数に0を指定すると"Illegal function call"のエラーになります。

CALL MUSIC(0,0,3,,,1)のようにチャンネル数指定を省略すると"Syntax error"になります。

パラメーターを省略したCALL MUSIC文

パラメーターをすべて省略して,

```
CALL MUSIC
```

のように実行すると,

```
CALL MUSIC(1,0,1,1,1)
```

が実行されたのと同じ結果になります。この場合のPLAY文の各サブコマンド列とFM音源, PSGとの対応は次のとおりです。

サブコマンド列	種類	出力
第1のサブコマンド列	メロディー	FM音源チャンネル1
第2のサブコマンド列	メロディー	FM音源チャンネル2
第3のサブコマンド列	メロディー	FM音源チャンネル3
第4のサブコマンド列	リズム	FM音源チャンネル7, 8, 9
第5, 6, 7のサブコマンド列	メロディー	PSG音源チャンネル1, 2, 3

メモリーの設定について

CALL MUSIC文を実行すると, FM音源処理用のメモリー領域を確保するため, BASICで使用するメモリー領域が初期値より807バイト分小さくなります。CLEAR文での上限番地設定は解除されます。また, CALL MUSIC文の実行によりすべての変数の内容は初期状態に戻ります。

♪ PITCH(pitch)

(ピッチ)

FM音源のメロディー音の高さを指定する。

形　式

CALL PITCH(ピッチ)　または

_PITCH(ピッチ)

ピッチ　条410≦ピッチ≦459

働きと使いかた

PLAY文のサブコマンド"O4A"に相当する音のピッチを調整します。"O4A"のピッチ(周波数)は，初期値では440Hzで，これを410Hzから459Hzの間で指定することができます。
CALL PITCH文で"O4A"のピッチを指定すると，それにともなってFM音源から出力される音全体のピッチが変化します。ただし，リズム音や音程を持たない音，PSG音源から出力される音の高さは常に一定で変化しません。
CALL PITCH文は，CALL TRANSPOSE文によるセント単位の移調からは独立して実行することができます。

実行例

```
CALL PITCH(450)
```

♪ STOPM(stop music)

ストップミュージック

バックグラウンドで実行中のPLAY文による演奏を停止する。

形　式

CALL STOPM　または

_STOPM

働きと使いかた

PLAY文による演奏がバックグラウンドで実行されているとき，演奏を停止します。

♪ TEMPER(temperament)

（テンペラメント）

FM音源の音律を指定する。

形　式

CALL TEMPER(音律)　または

_TEMPER(音律)

音律　条0≦音律≦21

働きと使いかた

初期状態では，PLAY文のメロディーは完全平均律で演奏されるようになっています。CALL TEMPER文で，音律を完全平均律以外に指定することができます。音律とは，1オクターブをどのような比率で12音に分割するかを決めるものです。

音律の指定	設定される音律
0	ピタゴラス
1	ミーントーン
2	ヴェルクマイスター
3	ヴェルクマイスター（修正1）
4	ヴェルクマイスター（修正2）
5	キルンベルガー
6	キルンベルガー（修正）
7	ヴァロッティ・ヤング
8	ラモー
9	完全平均律（初期設定）
10	純正律　cメジャー（aマイナー）
11	純正律　cisメジャー（bマイナー）
12	純正律　dメジャー（hマイナー）
13	純正律　esメジャー（cマイナー）
14	純正律　eメジャー（cisマイナー）

15	純正律　fメジャー（dマイナー）
16	純正律　fisメジャー（esマイナー）
17	純正律　gメジャー（eマイナー）
18	純正律　gisメジャー（fマイナー）
19	純正律　aメジャー（fisマイナー）
20	純正律　bメジャー（gマイナー）
21	純正律　hメジャー（gisマイナー）

実行例

```
CALL TEMPER(7)
```

♪ TRANSPOSE(transpose)

トランスポーズ

FM音源のメロディー音の高さをセント単位で移調する。

形　式

CALL TRANSPOSE(トランスポーズ値)　または

_TRANSPOSE(トランスポーズ値)

トランスポーズ値　条-12799≦トランスポーズ値≦12799。数値型定数，変数，配列変数，式。

働きと使いかた

セント単位で移調します。セントは半音を100とした移調の単位で，1セントは半音の100分の1の音階に相当します。したがって，1200セントで1オクターブになります。±12799セントの範囲で指定できますが，実際にはそのときのFM音源の音色によって音階の範囲は制限されます。また，FM音源用LSIの精度は±2セント程度です。

CALL TRANSPOSE文は，CALL PITCH文によるピッチの設定からは独立して実行することができます。

実行例

```
CALL TRANSPOSE(600)
```

半オクターブ高く移調します。

♪ VOICE(voice)

ボイス

FM音源の各チャンネルの音色を設定する。

形　式

CALL VOICE([チャンネル1用の音色],[チャンネル2用の音色],...,[チャンネル9用の音色])　または

_VOICE([チャンネル1用の音色],[チャンネル2用の音色],...,[チャンネル9用の音色])

音色　条@式（0≦式≦63）　または配列変数。
　　　省音色は変わらない。

働きと使いかた

MSXのFM音源システムには,64種類の音色が組み込まれており(ひとつは無音),CALL VOICE文により9つのチャンネルのそれぞれに違う音色を設定することができます。
初期状態では,すべてのチャンネルの音色は"ピアノ1"(音色番号@0)に設定されています。
すでに組み込まれている64種類の音色以外のものを作りたいときは,音色のデータを配列変数に入れておき,CALL VOICE文ではその配列変数名を指定します。

拡張 ♪

システムに組み込みの音色

音色番号	音色	音色番号	音色
@0	ピアノ1＊	@32	ピアノ3
@1	ピアノ2	@33	エレクトリックピアノ2＊
@2	バイオリン＊	@34	サンツール2
@3	フルート1＊	@35	ブラス
@4	クラリネット＊	@36	フルート2
@5	オーボエ＊	@37	クラビコード2
@6	トランペット＊	@38	クラビコード3
@7	パイプオルガン1	@39	琴2
@8	シロフォン（木琴）	@40	パイプオルガン2
@9	オルガン＊	@41	PohdsPLA
@10	ギター＊	@42	PohdsPRA
@11	サンツール1	@43	チャーチオルガンL
@12	エレクトリックピアノ1＊	@44	チャーチオルガンR
@13	クラビコード1	@45	シンセ・バイオリン
@14	ハープシコード1＊	@46	シンセ・オルガン
@15	ハープシコード2	@47	シンセ・ブラス
@16	ビブラホン＊	@48	ホルン＊
@17	琴1	@49	三味線
@18	太鼓	@50	マジカル
@19	エンジン1	@51	フワワ
@20	UFO	@52	ワンダーフラット
@21	シンセサイザーベル	@53	ハードロック
@22	チャイム	@54	マシン
@23	シンセ・ベース＊	@55	マシンV

拡張 ♪

@24	シンセサイザー＊	@56	コミック
@25	シンセ・パーカッション	@57	SE-コミック
@26	シンセ・リズム	@58	SE-レーザー
@27	ハーモ・ドラム	@59	SE-ノイズ
@28	カウベル	@60	SE-星1
@29	ハイハット	@61	SE-星2
@30	スネア・ドラム	@62	エンジン2
@31	バス・ドラム	@63	無音

音色の中で，＊記号のついていない音色どうしは同時に2つ以上設定することはできません。そのような設定をすると，CALL VOICE文で最後のチャンネル番号に設定した音色ひとつだけが有効になります。（同時に指定した＊のついた音色は，すべて有効になります。）

配列変数による音色の設定について

音色のデータは，ひとつの音色につき32バイトで構成されます。このデータをつくるには，32バイトのデータの意味を知る必要があります。データ構成については，参考をご覧ください。

なお，システムに組み込みの音色（@0から@62まで）を，VOICE COPY文で配列変数にコピーし，それから配列変数の値を変更することによって，組み込みの音色を変更して任意の音色を作るという方法もあります。

実行例

```
CALL VOICE(@0,@2,@4)
```

チャンネル1をピアノ1に，チャンネル2をバイオリンに，チャンネル3をクラリネットに設定します。

```
CALL VOICE(A%)
```

チャンネル1を，配列変数A%のデータで設定される音色にします。

```
CALL VOICE(@0,@7,@17)
```

チャンネル1はピアノ1に設定されますが，チャンネル2，3は琴1に設定されます。

```
CALL VOICE(@1,@18,A%)
```

チャンネル1，2，3とも配列変数A%のデータの音色に設定されます。

拡張 ♪

参考

音色データの構成

バイト1～8	音色名
バイト9～10	音色移調
バイト11	ビット0:無視される ビット1～3:フィードバック ビット4～7:無視される
バイト12～16	予約済み
バイト17	ビット0～3:MULT ビット4:KSR ビット5:EG ビット6:PM ビット7:AM
バイト18	ビット0～5:トータルレベル ビット6～7:レベルキースケール
バイト19	ビット0～3:ディケイレート ビット4～7:アタックレート
バイト20	ビット0～3:リリースレート ビット4～7:サスティンレベル
バイト21	無視される
バイト22～24	予約済み
バイト25～32	バイト17～24のくり返し

♪ VOICE COPY (voice copy)

ボイスコピー

音色のデータを転送する。

形　式

CALL VOICE COPY(音色1,音色2) または

_VOICE COPY(音色1,音色2)

音色1　条@式(0≦式≦63, ただし0, 2, 3, 4, 5, 6, 9, 10, 12, 14, 16, 23, 24, 33, 48を除く)　または配列変数名。

音色2　条@式 (式=63)　または配列変数名。

働きと使いかた

システムに組み込まれた音色のうち,231ページの表で＊のついていないものを@63の音色番号または配列変数に転送することができます。また,配列変数の音色データを@63の音色番号に転送することができます。

システムに組み込まれた音色を変更して新しい音色を作るときなどにVOICE COPY文が使えます。その手順は次のようになります。

1　32バイト分の配列変数を宣言する。(DIM A%(15)など)
2　変更したい音色を配列変数にコピーする。(CALL VOICE COPY(@7,A%)など)
3　配列変数の値を変更する。(新しい音色にする。)
4　変更後の配列変数のデータを@63にコピーする。(CALL VOICE COPY(A%, @63))
5　@63の音色をチャンネルに設定する。

拡張 ♪

♪ PLAY (play)

PLAY文が音楽を演奏中かどうかを調べる。

形　式

CALL PLAY(PLAY文のサブコマンド列番号, 数値変数)　または

_PLAY(PLAY文のサブコマンド列番号, 数値変数)

サブコマンド列番号　条0≦サブコマンド列番号≦12。

数値変数　条数値型の変数。

働きと使いかた

サブコマンド列番号で指定したサブコマンド列が演奏中であれば-1が，そうでなければ0が数値変数に代入されます。

サブコマンド列番号に0を指定すると，いずれかのサブコマンド列が演奏中であれば-1が，そうでなければ0が数値変数に代入されます。

サブコマンド列番号の最大値は，CALL MUSIC文で指定したFM音源のチャンネル数に3(PSGのチャンネル数）を加えたものです。

日本語入力を行うための拡張コマンド

PRINT文で漢字やかなを含む日本語の文字列を画面に表示したり，LPRINT文で日本語文字列をプリンターによって印字するために，次の拡張コマンドが準備されています。また，日本語文字列を操作するために便利な拡張コマンドも準備されています。

拡張コマンドは，いずれもCALL文により実行することができます。

漢 AKCNV（エーケーシーエヌブイ） (ank to kanji convert)

文字データをすべて全角文字に変換して，文字変数に代入する。

反 KACNV

形　式

CALL AKCNV（文字変数, X$） または

_AKCNV（文字変数, X$）

文字変数　条 文字型の変数。

X$　条 文字型定数，変数，配列変数，式。

働きと使いかた

X$に全角以外の文字が含まれていれば，それを全角文字に変換し，全体を文字変数に代入します。

実行例

```
CALL AKCNV(A$,"ABC漢字ｱｲｳｴｵ"):PRINT A$
ＡＢＣ漢字アイウエオ
```

CALL AKCNV文で与えられた文字列中，半角文字である"ABC"および"アイウエオ"が全角文字に変換されます。

漢 ANK（ank）

アンク

日本語入力モードを終了する。　　反 KANJI

形　式

CALL ANKまたは

_ANK

働きと使いかた

CALL KANJI文で設定した日本語入力モードを解除します。ただし，CALL KANJI文により確保された漢字ドライバー用のメモリーは解放されません。

漢 CLS（clear screen）

シーエルエス

日本語入力モードのスクリーン上のすべての表示を消去する。

形　式

CALL CLSまたは

_CLS

働きと使いかた

CALL KANJI文で日本語入力モードに設定してある状態のとき，画面をクリアするのに使用します。

日本語入力モードでCLS文を使用すると，正常に機能しない場合があります。

漢 JIS（jis）

ジス

文字データの最初の文字に対応するJISコードを文字変数に代入する。

反KNJ　関SJIS

形　式

CALL JIS(文字変数, X$)または

_CALL JIS(文字変数, X$)

文字変数　条文字型の変数。

X$　条文字型定数，変数，配列変数，式。

働きと使いかた

X$の先頭が2バイト型の文字であれば，そのJISコード（16進4けた）を文字型データとして文字変数に代入します。

実行例

```
CALL JIS(A$,"漢字"):PRINT A$
3441
```

漢 ## KACNV (kanji to ank convert)

ケーエーシーエヌブイ

文字データ中，半角文字に変換できるものを半角文字に変換し，文字変数に代入する。

反AKCNV

形　式

CALL KACNV(文字変数, X$)または

_KACNV(文字変数, X$)

文字変数　条文字型の変数。

X$　条文字型定数，変数，配列変数，式。

働きと使いかた

X$に含まれる文字を，次のように変換します。

変換前	変換後
全角のアルファベット，数字	半角のアルファベット，数字
全角のひらがな	半角のカタカナ
全角のカタカナ	半角のカタカナ
漢字	漢字（変換されない）

実行例

```
CALL KACNV(A$,"本日のお薦めメニュー"):PRINT A$
本日ﾉｵ薦ﾒﾆｭｰ
```

拡張
漢

漢 KANJI (kanji)

カンジ

日本語入力モードにする。　　反ANK

形　式

CALL KANJI[漢字フォント]または

_KANJI[漢字フォント]

漢字フォント　条0≦モード≦3。数値型定数，変数，配列変数，式。
省0。

働きと使いかた

CALL KANJI文を実行すると日本語入力モードになります。解除するときはCALL ANK文を実行します。

漢字フォントの選択

漢字フォントの値	使用される漢字フォント
0	横16ドット X たて16ドット
1	MSXの標準漢字ROMを使用。 横12ドット X たて16ドットに圧縮。
2	漢字フォント0のときと同じ。 インターレースモードにより，たて方向が圧縮され，表示行数が増える。
3	漢字フォント1のときと同じ。 インターレースモードにより，たて方向が圧縮され，表示行数が増える。

スクリーンモードについて

CALL KANJI 0またはCALL KANJI 2を実行したとき，WIDTH文による1行の文字数が20から32の場合はSCREEN 5のモードに，1行の文字数が33から64の場合はSCREEN 7のモードに設定されます。
CALL KANJI 1またはCALL KANJI 3を実行したとき，WIDTH文による1行の文字数が20から40の場合はSCREEN 5のモードに，1行の文字数が41から80の場合はSCREEN 7のモードに設定されます。

メモリーの設定について

MSX_{2+}-BASICをスタートしてから最初にCALL KANJI文を実行したときは，日本語処理用のメモリー領域を確保する処理が行われます。このとき，CLEAR文での上限番地の設定は解除され，すべての変数およびスタック（FOR-NEXT文，GOSUB/RETURN文用）の内容も初期状態に戻ります。

日本語入力の方法

MSX-JEが組み込まれていない場合は単漢字変換機能が，MSX-JEが組み込まれている場合は連文節変換機能が使えます。詳しくは，第1章の「日本語の入力」を参照してください。

漢 KEXT(ケーエクスト) (kanji extract)

文字データの中から，全角文字だけあるいは半角文字だけを抜き出し，文字変数に代入する。

関KMID

形　式

CALL KEXT(文字変数, X$, 機能)または

_KEXT(文字変数, X$, 機能)

文字変数	条	文字型の変数。
X$	条	文字型定数，変数，配列変数，式。
機能	条	0または1。

働きと使いかた

機能の値が0のときは文字データの中の半角文字だけを，1のときは全角文字だけを抜き出して，文字変数に代入します。

実行例

```
CALL KEXT(A$,"最初はｸﾞｰ",0):PRINT A$
ｸﾞｰ

CALL KEXT(A$,"最初はｸﾞｰ",1):PRINT A$
最初は
```

拡張
漢

漢 KINSTR (kanji in string)

ケーインストリング

文字データの中から指定した文字データを検索し，その位置を数値型変数に代入する。

形　式

CALL KINSTR(数値変数 [, N],X$, Y$) または

_KINSTR(数値変数 [, N],X$, Y$)

数値変数　条数値型の変数。

N　条1≦N＜256。数値型定数，変数，配列変数，式。

省1。

X$，Y$　条文字型定数，変数，配列変数，式。

働きと使いかた

文字列X$の中にあるY$が，左から何番目から始まるかを調べ，その値を数値型変数に代入します。Nを指定すると，X$のN番目の文字から検索を開始します。

文字列が日本語入力によるものの場合，INSTR関数の代わりにCALL KINSTR文を使います。

実行例

```
CALL KINSTR(A,"TOKYO東京ﾄｳｷｮｳ","ﾄ"):PRINT A
 8
```

● Nの値がX$の長さより大きいとき，またはX$がヌルストリングのとき，Y$が見つからないときは数値変数には0が代入されます。また，Y$がヌルストリングのとき数値変数には1が代入されます。(ただし，X$もヌルストリングのときは0が代入されます。)

漢 ケーレングス
KLEN (kanji length)

文字データの全体の文字数(長さ), または半角文字だけの文字数, または全角文字だけの文字数を数値型変数に代入する。

形 式

CALL KLEN(数値変数, X$[, 機能]) または

_KLEN(数値変数, X$[, 機能])

数値変数	条数値型の変数。
X$	条文字型定数, 変数, 配列変数, 式。
機能	条0≦機能≦2。
	省0

働きと使いかた

機能の値が0(または省略)のときは文字データ全体の文字数(長さ)を, 1のときは半角文字の文字数を, 2のときは全角文字の文字数を数値変数に代入します。

文字データが日本語入力によるものの場合, LEN関数の代わりにCALL KLEN文を使います。

実行例

```
CALL KLEN(A,"雨ニモマケズ"):PRINT A
 7
CALL KLEN(A,"雨ニモマケズ",1):PRINT A
 6
CALL KLEN(A,"雨ニモマケズ",2):PRINT A
 1
```

●半角の濁点も1文字として数えます。

拡張 漢

漢 KMID（kanji middle）

文字データの一部分を抜き出して文字変数に代入する。

関KEXT

形　式

CALL KMID(文字変数, X$, M[, N]) または

_KMID(文字変数, X$, M[, N])

文字変数	条文字型の変数。
X$	条文字型定数，変数，配列変数，式。
M	条1≦M＜256。数値型定数，変数，配列変数，式。
N	条0≦N＜256。数値型定数，変数，配列変数，式。
	省M字目以降の全文字を代入する。

働きと使いかた

文字データのM番目からN字分だけ抜き出して文字変数に代入します。

文字データが日本語入力によるものの場合，MID$関数の代わりにCALL KMID文を使います。

実行例

```
CALL KMID(A$,"今日は暑いですか",4,2):PRINT A$
暑い
```

●半角の濁点も1文字として数えます。

拡張 漢

漢 KNJ (kanji)

カンジ

指定した漢字コードに対応する文字を文字変数に代入する。

反JIS, SJIS

形　式

CALL KNJ(文字変数, X$) または

_KNJ(文字変数, X$)

文字変数　条文字型の変数。

X$　条4桁の漢字コードを表わす文字型定数, 変数, 配列変数, 式。

働きと使いかた

X$で16進4けたの漢字コードを指定します。漢字コードが8000未満の場合はJISコード, 8000以上の場合はシフトJISコードとみなされます。

実行例

```
CALL KNJ(A$,"3441"):PRINT A$
漢
```

漢 KTYPE (kanji type)

ケータイプ

文字データの中の指定した位置の文字が半角か全角かを調べる。

形　式

CALL KTYPE(数値変数, X$, N)　または

_KTYPE(数値変数, X$, N)

数値変数	条数値型の変数。
X$	条文字型定数，変数，配列変数，式。
N	条0＜N＜256。数値型定数，変数，配列変数，式。

働きと使いかた

X$のN番目の文字が半角なら0を，全角なら1を数値変数に代入します。

NをX$の文字数より大きく指定すると"Illegal function call"のエラーになります。

実行例

```
CALL KTYPE(A,"ﾊﾝｶｸ全角",1):PRINT A
 0
CALL KTYPE(A,"ﾊﾝｶｸ全角",5):PRINT A
 1
```

●半角の濁点も1文字として数えます。

漢 PALETTE（palette）

パレット

日本語入力モードのとき，パレット番号に表示される色を割り当てる。

形　式

CALL PALETTE(パレット番号, 赤の輝度, 緑の輝度, 青の輝度）または

_PALETTE(パレット番号, 赤の輝度, 緑の輝度, 青の輝度)

パレット番号　　条0≦パレット番号＜16。数値型定数，変数，配列変数，式。

赤，緑，青の輝度　条0≦輝度≦7。整数型定数。

働きと使いかた

CALL KANJI文を実行して日本語入力モードに設定してあるとき，各パレット番号の色を決めるために使います。（COLOR＝文は使えません。）

パラメーターをすべて省略すると，パレットは初期状態に戻ります。

実行例

```
CALL PALETTE(15, 4, 4, 4)
```

拡張 漢

漢 SJIS（shift jis）

エスジス

文字データの最初の文字に対応するシフトJISコードを文字変数に代入する。

反KNJ, 関JIS

形　式

CALL SJIS(文字変数, X$) または

_CALL SJIS(文字変数, X$)

文字変数　条文字型の変数

X$　条文字型定数, 変数, 配列変数, 式

働きと使いかた

X$の先頭が2バイト型の文字であれば, そのシフトJISコード (16進4けた) を文字型データとして文字変数に代入します。

実行例

```
CALL SJIS(A$,"漢字"):PRINT A$
8ABF
```

拡張 漢

エラーメッセージ

エラーが発生すると，プログラムの実行が止まり，コマンド待ちの状態に戻るとともにエラーメッセージが表示されます。エラーメッセージは，そのエラーの原因を簡潔に示しています。以下にエラーメッセージと，その原因の具体例を挙げておきます。エラーメッセージの次の（　）の中の数字は，そのエラー番号を示します。

Bad FAT（60）

●ディスクがフォーマットされていない。

Bad drive name（62）

●デバイス名で，使われていないディスクドライブを指定した。

Bad file mode（61）

●シーケンシャルファイルに対してPUT文，GET文，LOF関数を使った。

●ランダムアクセスファイルに対してLOAD文が実行された。

●OPEN文のモードの指定が不適当。

Bad file name（56）

●ファイル名が不適当。

●OPEN文やSAVE文，LOAD文などで，指定できないデバイス名を指定した。

Bad file number（52）

●MAXFILES＝文で指定した範囲以上のファイル番号を使った。

●開いていないファイルの番号を使ってPRINT#文などを実行しようとした。

Bad sector number（63）

●PUT文，GET文で指定したレコード番号が0，または32767より大きい。

Can't CONTINUE（17）

●中断の後，プログラムを変更してから再開しようとした。

●プログラムが存在しない。

●プログラム中でCONT文を使用した。

Device I/O error (19)

- カセットテープまたはテープレコーダーが原因でうまくロードできなかった。
- テープレコーダーのレベルが不適当。
- 完全にロードし終わらないうちにコマンドを中断した。
- その他入出力装置のエラー。

Direct statement in file (57)

- ロード中のアスキー形式のプログラム中に行番号のない文があった。
- BASICプログラム以外のファイル（データファイルなど）をロードしようとした。

Disk full (66)

- ディスクの残り容量がなくなった。

Disk I/O error (69)

- ディスクとの入出力時に，回復不可能なエラーが生じた。

Disk offline (70)

- ディスクドライブがつながっていない。

Disk write protected (68)

- 書き込み禁止のディスクに対して書き込みを行った。

Division by zero (11)

- ゼロで割る割り算を実行しようとした。
- 定義されていない変数で割る割り算を実行しようとした。

FIELD overflow (50)

- FIELD文での領域の指定が，レコードの長さを越えた。

File alredy exists (65)

- NAME文で新しいファイル名として指定したファイル名が，すでにディスク上に存在する。

File already open (54)

- すでに開かれているファイルを再度開こうとした。
- すでに開かれているファイルにKILLコマンドを使おうとした。

File not found（53）

●LOAD，KILL，OPEN文で指定したファイルがディスク上にない。

Fiie not OPEN（59）

●まだOPEN文で開いていないファイル番号を使ってPRINT#文，INPUT#文などを実行しようとした。

Fiie still open（64）

●ファイルのCLOSEがされていない。

Illegal direct（12）

●DEFFN文など，プログラム中でしか使えない文を直接コマンドモードで実行しようとした。

Illegal function call（5）

●コマンドへ引き渡す値が間違っている。

●関数へ引き渡す値が許された範囲の外にある。

Input past end（55）

●ファイルのデータがすべて読み込まれているのに，さらに読み込もうとした。

●ファイルにデータがない。

Internal error（51）

●BASICインタープリンターに異常がある。

Line buffer overflow（25）

●入力ラインバッファーが一杯になった。

Missing operand（24）

●コマンドに続くべきパラメーターがない。

●必要なパラメーターがそろっていない。

NEXT without FOR（1）

●実行されたNEXT文に対応するFOR文がない。

●FOR-NEXTループ中にGOTO文などで実行を移した。

NO RESUME (21)

●エラー処理ルーチンにRESUME文がない。
(エラー処理ルーチンはEND, RESUME, ON ERROR GOTO 0のいずれかで終わらねばなりません)

Out of DATA (4)

●READ文で読むべきデータがない，または足りない。

Out of memory (7)

●プログラムが長すぎる。
●使用する変数が多すぎる。
●配列が大きすぎる。
●FOR-NEXT文，GOSUB-RETURN文での多重構造が深すぎる。

Out of string space (14)

●文字領域が足りない。
●CLEAR文で指定した文字領域が小さすぎる。

Overflow (6)

●数値型データや演算結果が，扱える範囲を起えた。
●パラメーターとして与えるアドレスが，指定範囲を外れた。

Rename across disk (71)

●異なるディスクドライブ間でNAME文を実行しようとした。

RAM disk full (66)

●メモリーディスクの残り容量がなくなった。(ただし，Disk-BASICではdisk fullとでます。)

RAM disk offline (70)

●CALL MEMINIが実行されていないのに，メモリーディスクを使おうとした。(ただし，Disk-BASICではdisk offlineとでます。)

RESUME without error (22)

●RESUME文に対応するON ERROR文がない。
●エラー処理ルーチンにGOTO文などで移行した。
●メインルーチンの最後にEND文などがないため，引き続いてサブルーチンが実行された。

RETURN without GOSUB (3)

●RETURN文に対応するGOSUB文がない。

●サブルーチンにGOTO文などで移行した。

●メインルーチンの最後にEND文などがないため，引き続いてサブルーチンが実行された。

Redimensioned array (10)

●同じ名前の配列に2重に定義しようとした。

●DIM文で宣言せずに配列変数を使い，その後で定義した。

Sequential I/O only (58)

●シーケンシャルファイルに対してランダムアクセスファイル用のコマンドを使った。

String formula too complex (16)

●1行に書かれた文字式が複雑すぎる。

String too long (15)

●文字変数に255文字以上代入しようとした。

Subscript out of range (9)

●DIM文で宣言した大きさを超える添字を使った。

●DIM文で宣言していない配列変数に対して，11以上の添字を使った。

Syntax error (2)

●入力された文が，MSX2-BASICの文法に合わない。

Too many files (67)

●ファイルの数が112を越えた。

●ファイルの数が32（メモリーマッパー対応の場合は64）を越えた。

Type mismatch (13)

●代入文などで，左辺と右辺の型が合っていない。

●文字型データを論理演算した。

●関数の指定したデータの型が合っていない。

Undefined　line　number (8)

●GOTO文やGOSUB文，RESUME文などで，存在しない行番号を指定した。

●RENUM文を実行した際，プログラム中のGOTO文などが存在しない行番号を指定していた。

Undefined　user　function (18)

●DEFFN文で定義していないユーザー関数を使おうとした。

Unprintable　error (23, 26-49, 72-255)

●エラー番号を持たないエラーが発生した。

●ERROR文で，Unprintable　errorのエラー番号を指定してエラーを発生させた。

Verify　error (20)

●カセットテープのプログラムとメモリー上のプログラムとが異なっている。

付録

文字

MSX$_{2+}$-BASICで扱える文字

MSX$_{2+}$-BASICがPRINT文などにより表示することのできる文字とその文字コードは以下の通りです。

16進数	00-1F		20-3F		40-5F		60-7F		80-9F		A0-BF		C0-DF		E0-FF	
	コード	文字	コード	文字	コード	文字	コード	文字	コード	文字	コード	文字	コード	文字	コード	文字
0	0		32		64	@	96	`	128	♠	160		192	タ	224	た
1	1		33	!	65	A	97	a	129	♥	161	。	193	チ	225	ち
2	2		34	"	66	B	98	b	130	♣	162	「	194	ツ	226	つ
3	3		35	#	67	C	99	c	131	♦	163	」	195	テ	227	て
4	4		36	$	68	D	100	d	132	○	164	、	196	ト	228	と
5	5		37	%	69	E	101	e	133	●	165	・	197	ナ	229	な
6	6		38	&	70	F	102	f	134	を	166	ヲ	198	ニ	230	に
7	7		39	'	71	G	103	g	135	ぁ	167	ァ	199	ヌ	231	ぬ
8	8		40	(	72	H	104	h	136	ぃ	168	ィ	200	ネ	232	ね
9	9		41	)	73	I	105	i	137	ぅ	169	ゥ	201	ノ	233	の
A	10		42	*	74	J	106	j	138	ぇ	170	ェ	202	ハ	234	は
B	11		43	+	75	K	107	k	139	ぉ	171	ォ	203	ヒ	235	ひ
C	12		44	,	76	L	108	l	140	ゃ	172	ャ	204	フ	236	ふ
D	13		45	-	77	M	109	m	141	ゅ	173	ュ	205	ヘ	237	へ
E	14		46	.	78	N	110	n	142	ょ	174	ョ	206	ホ	238	ほ
F	15		47	/	79	O	111	o	143	っ	175	ッ	207	マ	239	ま
0	16		48	0	80	P	112	p	144		176	ー	208	ミ	240	み
1	17		49	1	81	Q	113	q	145	あ	177	ア	209	ム	241	む
2	18		50	2	82	R	114	r	146	い	178	イ	210	メ	242	め
3	19		51	3	83	S	115	s	147	う	179	ウ	211	モ	243	も
4	20		52	4	84	T	116	t	148	え	180	エ	212	ヤ	244	や
5	21		53	5	85	U	117	u	149	お	181	オ	213	ユ	245	ゆ
6	22		54	6	86	V	118	v	150	か	182	カ	214	ヨ	246	よ
7	23		55	7	87	W	119	w	151	き	183	キ	215	ラ	247	ら
8	24		56	8	88	X	120	x	152	く	184	ク	216	リ	248	り
9	25		57	9	89	Y	121	y	153	け	185	ケ	217	ル	249	る
A	26		58	:	90	Z	122	z	154	こ	186	コ	218	レ	250	れ
B	27		59	;	91	[	123	{	155	さ	187	サ	219	ロ	251	ろ
C	28		60	<	92	¥	124	¦	156	し	188	シ	220	ワ	252	わ
D	29		61	=	93	]	125	}	157	す	189	ス	221	ン	253	ん
E	30		62	>	94	^	126	~	158	せ	190	セ	222	゛	254	
F	31		63	?	95	-	127		159	そ	191	ソ	223	゜	255	■

文字コードが2バイトのもの

次のグラフィックキャラクターは，先頭が01H，2番目が表に示されるコードの2バイトのコードから成ります。

16進数	40-4F		50-5F	
	コード	文字	コード	文字
0	64		80	π
1	65	月	81	┴
2	66	火	82	┬
3	67	水	83	┤
4	68	木	84	├
5	69	金	85	┼
6	70	土	86	│
7	71	日	87	─
8	72	年	88	┌
9	73	円	89	┐
A	74	時	90	└
B	75	分	91	┘
C	76	秒	92	╳
D	77	百	93	大
E	78	千	94	中
F	79	万	95	小

文字コードの入出力

キーボードから入力したとき

通常の文字…1バイトのコードを入力

例：Aはコード65（10進数。16進では41H）

2 バイトコードの文字…1ともう1バイトのコード入力

例…月はコード1，65（10進数。16進では1H，41H）

CHR$関数を使って出力するとき

通常の文字…1バイトのコードをパラメーターとする

例…BはCHR$（66）（またはCHR$（&H42））

2バイトのコードの文字…最初はCHR$（1），次に表のコードをパラメーターとする。

例：火はCHR$（1）＋CHR$（66）

（またはCHR$（1）＋CHR$（&H42））

JISコード表(漢字外)

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
記号(NORMAL)	2120			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
	2130	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／
	2140	＼	～	∥	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
	2150	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
	2160	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
	2170	＄	¢	£	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	
	2220		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓	
特殊記号	2220																㈱
	2230	㈲	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	g	m	cm	mm	km
	2240	cm²	m²	mℓ	mg	kg	μs	㈹	KK	TEL	No.	㌅		㌘	㍍	㌢	㍉
	2250	㌶	㌃	㌦	㌧	㌔	㌕	㌗	㍗	㍑	㌫	㌻	㍎	㌳	①	②	③
	2260	④	⑤	⑥	⑦	⑧	⑨	⑩	〚	〛	☏	☎	[illegible]	∫	√	￣	⊓
	2270	¦	｜	⊔	⌒	◡	［	＝	］	（	）	(CR1)	(CR2)	㊗	㊐	(↑)	(↓)
	2320		◂—	◀	‥‥	‥	·	：	⇨								
英・数字(NORMAL)	2330	0	1	2	3	4	5	6	7	8	9						
	2340		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	p	q	r	s	t	u	v	w	x	y	z					
ひらがな(標準)	2420		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	2460	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	ゐ	ゑ	を	ん	▨	▧	▥	▤	░							
カタカナ(標準)	2520		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	2560	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									
ギリシャ文字	2620		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	2630	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	2640	░	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	2650	π	ρ	σ	τ	υ	φ	χ	ψ	ω							▦

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ロシア文字	2720		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
	2730	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	2740	Ю	Я														
	2750		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	2760	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	2770	ю	я														

JIS第1水準漢字コード表

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
あ	3020		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
い	3040											以	伊	位	依	偉	囲
	3050	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120		院	陰	隠	韻	吋										
う	3120							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	雲															
え	3140		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	艶	苑	薗	遠	鉛	鴛	塩									
お	3170								於	汚	甥	凹	央	奥	往	応	
	3220		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
か	3230													下	化	仮	何
	3240	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	3350	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	3370	叶	椛	樺	鞄	株	兜	竃	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
き	3460												企	伎	危	喜	器
	3470	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	3560	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	3620		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	謹	近	金	吟	銀											
く	3660						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	3670	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	3730	薫	訓	群	軍	郡											
け	3730						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	3760	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	言	諺	限													
こ	3840				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	3940	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降
	3960	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A20		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	紺	艮	魂													
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
さ	3A30				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	酸	餐	斬	暫	残											
し	3B40						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C60	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D30	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D50	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E30	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E70	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						
す	3F50											笥	諏	須	酢	図	厨
	3F60	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020		澄	摺	寸												
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
せ	4020					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	4120		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
	4130	前	善	漸	然	全	禅	繕	膳	糎							
そ	4130										噌	塑	岨	措	曾	曽	楚
	4140	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	4220		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
た	4230															他	多
	4240	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
ち	4340														値	知	地
	4350	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	4420		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	沈	珍	賃	鎮	陳											
つ	4440						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	4460	釣	鶴														
て	4460			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	4470	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	4540	点	伝	殿	澱	田	電										
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
と	4540							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬
	4560	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟
	4570	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	
	4620		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
な	4660	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	4670	軟	難	汝													
に	4670				二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	
	4720		如	尿	韮	任	妊	忍	認								
ぬ	4720									濡							
ね	4720										禰	祢	寧	葱	猫	熱	年
	4730	念	捻	撚	燃	粘											
の	4730						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	農	覗	蚤													
は	4740				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	4760	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
	4770	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	4840	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ひ	4850												匪	卑	否	妃	庇
	4860	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	賓	頻	敏	瓶												
ふ	4950					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A30	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
へ	4A30											丙	併	兵	塀	幣	平
	4A40	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ほ	4A50														保	舗	鋪
	4A60	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B30	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B50	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
ま	4B60	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B70	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C20		漫	蔓													
み	4C20				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	粍	民	眠													
む	4C30				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
め	4C30														冥	名	命
	4C40	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
も	4C40															摸	模
	4C50	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	尤	戻	籾	貰	問	悶	紋	門	匁							
や	4C60										也	冶	夜	爺	耶	野	弥
	4C70	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ゆ	4C70												愉	愈	油	癒	
	4D20		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D30	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
よ	4D30														予	余	与
	4D40	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
	4D60	沃	浴	翌	翼	淀											
ら	4D60						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
	4D70	乱	卵	嵐	欄	濫	藍	蘭	覧								
り	4D70									利	吏	履	李	梨	理	璃	
	4E20		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
る	4E50													瑠	塁	涙	累
	4E60	類															
れ	4E60		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20		蓮	連	錬												
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ろ	4F20					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録
	4F40	論															
わ	4F40		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	椀	湾	碗	腕												
	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

JIS第2水準漢字コード表

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
一	5020		弌	丐	丕												
丨	5020					个	丱										
丶	5020							丶	丼								
丿	5020									丿	乂	乖	乘				
乙	5020													亂			
亅	5020														亅	豫	亊
	5030	舒															
二	5030		弍	于	亞	亟											
亠	5030						亠	亢	亰	亳	亶						
人	5030											从	仍	仄	仆	仂	仗
	5040	仞	仭	仟	价	伉	佚	估	佛	佝	佗	佇	佶	侈	侏	侘	佻
	5050	佩	佰	侑	佯	來	侖	儘	俔	俟	俎	俘	俛	俑	俚	俐	俤
	5060	俥	倚	倨	倔	倪	倥	倅	伜	俶	倡	倩	倬	俾	俯	們	倆
	5070	偃	假	會	偕	偐	偈	做	偖	偬	偸	傀	傚	傅	傴	傲	
	5120		僉	僊	傳	僂	僖	僞	僥	僭	僣	僮	價	僵	儉	儁	儂
	5130	儖	儕	儔	儚	儡	儺	儷	儼	儻							
儿	5130										儿	兀	兒	兌	兔	兢	竸
入	5140	兩	兪														
八	5140			兮	冀												
冂	5140					冂	囘	册	冉	冏	冑	冓	冕				
冖	5140													冖	冤	冦	冢
	5150	冩	冪														
冫	5150			冫	决	冱	冲	冰	况	冽	凅	凉	凛				
几	5150													几	處	凩	凭
	5160	凰															
凵	5160		凵	凾													
刀	5160				刄	刋	刔	刎	刧	刪	刮	刳	刹	剏	剄	剋	剌
	5170	剞	剔	剪	剴	剩	剳	剿	剽	劍	劔	劒	剱	劈	劑	辧	
	5220		辨														
力	5220			劬	劭	劼	劵	勁	勍	勗	勞	勣	勦	飭	勠	勳	勵
	5230	勸															
勹	5230		勹	匆	匈	甸	匍	匐	匏								
匕	5230									匕							
匚	5230										匚	匣	匯	匱	匳		
匸	5230															匸	區
十	5240	卆	卅	丗	卉	卍	凖										
卜	5240							卞									
卩	5240								卩	卮	夘	卻	卷				
厂	5240													厂	厖	厠	厦
	5250	厥	厮	厰													
厶	5250				厶	參	簒										

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
又	5250							雙	叟	曼	燮						
口	5250											叮	叨	叭	叺	吁	吽
	5260	呀	听	吭	吼	吮	吶	吩	吝	呎	咏	呵	咎	呟	呱	呷	呰
	5270	咒	呻	咀	呶	咄	咐	咆	哇	咢	咸	咥	咬	哄	哈	咨	
	5320		咫	哂	咤	咾	咼	哘	哥	哦	唏	唔	哽	哮	哭	哺	哢
	5330	唹	啀	啣	啌	售	啜	啅	啖	啗	唸	唳	啝	喙	喀	咯	喊
	5340	喟	啻	啾	喘	喞	單	啼	喃	喩	喇	喨	嗚	嗅	嗟	嗄	嗜
	5350	嗤	嗔	嘔	嗷	嘖	嗾	嗽	嘛	嗹	噎	噐	營	嘴	嘶	嘲	嘸
	5360	噫	噤	嘯	噬	噪	嚆	嚀	嚊	嚠	嚔	嚏	嚥	嚮	嚶	嚴	囂
	5370	嚼	囁	囃	囀	囈	囎	囑	囓								
囗	5370									囗	囮	囹	圀	囿	圄	圉	
	5420		圈	國	圍	圓	團	圖	嗇	圜							
土	5420										圦	圷	圸	坎	圻	址	坏
	5430	坩	埀	垈	坡	坿	垉	垓	垠	垳	垤	垪	垰	埃	埆	埔	埒
	5440	埓	堊	埖	埣	堋	堙	堝	塲	堡	塢	塋	塰	毀	塒	堽	塹
	5450	墅	墹	墟	墫	墺	壞	墻	墸	墮	壅	壓	壑	壗	壙	壘	壥
	5460	壜	壤	壟													
士	5460				壯	壺	壹	壻	壼	壽							
夂	5460										夂						
夊	5460											夊	夐				
夕	5460													夛	梦	夥	
大	5460																夬
	5470	夭	夲	夸	夾	竒	奕	奐	奎	奚	奘	奢	奠	奧	奬	奩	
女	5520		奸	妁	妝	佞	侫	妣	妲	姆	姨	姜	妍	姙	姚	娥	娟
	5530	娑	娜	娉	娚	婀	婬	婉	娵	娶	婢	婪	媚	媼	媾	嫋	嫂
	5540	媽	嫣	嫗	嫦	嫩	嫖	嫺	嫻	嬌	嬋	嬖	嬲	嫐	嬪	嬶	嬾
	5550	孃	孅	孀													
子	5550				孑	孕	孚	孛	孥	孩	孰	孳	孵	學	斈	孺	
宀	5550																宀
	5560	它	宦	宸	寃	寇	寉	寔	寐	寤	實	寢	寞	寥	寫	寰	寶
	5570	寳															
寸	5570		尅	將	專	對											
小	5570						尓	尠									
尢	5570								尢	尨							
尸	5570										尸	尹	屁	屆	屎	屓	
	5620		屐	屏	孱	屬											
屮	5620						屮										
山	5620							乢	屶	屹	岌	岑	岔	妛	岫	岻	岶
	5630	岼	岷	峅	岾	峇	峙	峩	峽	峺	峭	嶌	峪	崋	崕	崗	嵜
	5640	崟	崛	崑	崔	崢	崚	崙	崘	嵌	嵒	嵎	嵋	嵬	嵳	嵶	嶇
	5650	嶄	嶂	嶢	嶝	嶬	嶮	嶽	嶐	嶷	嶼	巉	巍	巓	巒	巖	

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
巛	5650																巛
工	5660	巫															
己	5660		已	巵													
巾	5660				帋	帚	帙	帑	帛	帶	帷	幄	幃	幀	幎	幗	幔
	5670	幟	幢	幤	幇												
干	5670					幵	并										
幺	5670							幺	麼								
广	5670									广	庠	廁	廂	廈	廐	廏	
	5720		廖	廣	廝	廚	廛	廢	廡	廨	廩	廬	廱	廳	廰		
廴	5720															廴	廸
廾	5730	廾	弃	弉	彝	彜											
弋	5730						弋	弑									
弓	5730								弖	弩	弭	弸	彁	彈	彌	彎	弯
彐	5740	彑	彖	彗	彙												
彡	5740					彡	彭										
彳	5740							彳	彷	徃	徂	彿	徊	很	徑	徇	從
	5750	徙	徘	徠	徨	徭	徼										
心	5750							忖	忻	忤	忸	忱	忝	悳	忿	怡	恠
	5760	怙	怐	怩	怎	怱	怛	怕	怫	怦	怏	怺	恚	恁	恪	恷	恟
	5770	恊	恆	恍	恣	恃	恤	恂	恬	恫	恙	悁	悍	惧	悃	悚	
	5820		悄	悛	悖	悗	悒	悧	悋	惡	悸	惠	惓	悴	忰	悽	惆
	5830	悵	惘	慍	愕	愆	惶	惷	愀	惴	惺	愃	愡	惻	惱	愍	愎
	5840	慇	愾	愨	愧	慊	愿	愼	愬	愴	愽	慂	慄	慳	慷	慘	慙
	5850	慚	慫	慴	慯	慥	慱	慟	慝	慓	慵	憙	憖	憇	憬	憔	憚
	5860	憊	憑	憫	憮	懌	懊	應	懷	懈	懃	懆	憺	懋	罹	懍	懦
	5870	懣	懶	懺	懴	懿	懽	懼	懾	戀							
戈	5870										戈	戉	戍	戌	戔	戛	
	5920		戞	戡	截	戮	戰	戲	戳								
戸	5920									扁							
手	5920										扎	扞	扣	扛	扠	扨	扼
	5930	抂	抉	找	抒	抓	抖	拔	抃	抔	拗	拑	抻	拏	拿	拆	擔
	5940	拈	拜	拌	拊	拂	拇	抛	拉	挌	拮	拱	挧	挂	挈	拯	拵
	5950	捐	挾	捍	搜	捏	掖	掎	掀	掫	捶	掣	掏	掉	掟	掵	捫
	5960	捩	掾	揩	揀	揆	揣	揉	插	揶	揄	搖	搴	搆	搓	搦	搶
	5970	攝	搗	搨	搏	摧	摯	摶	摎	攪	撕	撓	撥	撩	撈	撼	
	5A20		據	擒	擅	擇	撻	擘	擂	擱	擧	舉	擠	擡	抬	擣	擯
	5A30	攬	擶	擴	擲	擺	攀	擽	攘	攜	攅	攤	攣	攫			
支	5A30														攴	攵	攷
	5A40	收	攸	畋	效	敖	敕	敍	敘	敞	敝	敲	數	斂	斃	變	
斗	5A40																斛
	5A50	斟															

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
斤	5A50		斫	斷													
方	5A50				旃	旆	旁	旄	旌	旒	旛	旙	无				
旡	5A50													旡			
日	5A50														旱	杲	昊
	5A60	昃	旻	杳	昵	昶	昴	昜	晏	晄	晉	晁	晞	晝	晤	晧	晨
	5A70	晟	晢	晰	暃	暈	暎	暉	暄	暘	暝	曁	暹	曉	暾	暼	
	5B20		曄	暸	曖	曚	曠	昿	曦	曩							
曰	5B20										曰	曳	曷				
月	5B20													朏	朖	朞	朦
	5B30	朧	霸														
木	5B30			朮	朿	朶	杁	朸	朷	杆	杞	杠	杙	杣	杤	枉	杰
	5B40	枩	杼	杪	枌	枋	枦	枡	枅	枷	柯	枴	柬	枳	柩	枸	柤
	5B50	柞	柝	柢	柮	枹	柎	柆	柧	檜	栞	框	栩	桀	桍	栲	桎
	5B60	梳	栫	桙	档	桷	桿	梟	梏	梭	梔	條	梛	梃	檮	梹	桴
	5B70	梵	梠	梺	椏	梍	桾	椁	棊	椈	棘	椢	椦	棡	椌	棍	
	5C20		棔	棧	棕	椶	椒	椄	棗	棣	椥	棹	棠	棯	椨	椪	椚
	5C30	椣	椡	棆	楹	楷	楜	楸	楫	楔	楾	楮	椹	楴	椽	楙	椰
	5C40	楡	楞	楝	榁	楪	榲	榮	槐	榿	槁	槓	榾	槎	寨	槊	槝
	5C50	榻	槃	榧	樮	榑	榠	榜	榕	榴	槞	槨	樂	樛	槿	權	槹
	5C60	槲	槧	樅	榱	樞	槭	樔	槫	樊	樒	櫁	樣	樓	橄	樌	橲
	5C70	樶	橸	橇	橢	橙	橦	橈	樸	樢	檐	檍	檠	檄	檢	檣	
	5D20		檗	蘗	檻	櫃	櫂	檸	檳	檬	櫞	櫑	櫟	檪	櫚	櫪	櫻
	5D30	欅	蘖	櫺	欒	欖	鬱	欟									
欠	5D30								欸	欷	盜	欹	飮	歇	歃	歉	歐
	5D40	歙	歔	歛	歟	歡											
止	5D40						歸										
歹	5D40							歹	歿	殀	殄	殃	殍	殘	殕	殞	殤
	5D50	殪	殫	殯	殲	殱											
殳	5D50						殳	殷	殼	毆							
母	5D50										毋	毓					
毛	5D50												毟	毬	毫	毳	毯
	5D60	麾	氈														
氏	5D60			氓													
气	5D60				气	氛	氤	氣									
水	5D60								汞	汕	汢	汪	沂	沍	沚	沁	沛
	5D70	汾	汨	汳	沒	沐	泄	泱	泓	沽	泗	泅	泝	沮	沱	沾	
	5E20		沺	泛	泯	泙	泪	洟	衍	洶	洫	洽	洸	洙	洵	洳	洒
	5E30	洌	浣	涓	浤	浚	浹	浙	涎	涕	濤	涅	淹	渕	渊	涵	淇
	5E40	淦	涸	淆	淬	淞	淌	淨	淒	淅	淺	淙	淤	淕	淪	淮	渭
	5E50	湮	渮	渙	湲	湟	渾	渣	湫	渫	湶	湍	渟	湃	渺	湎	渤
	5E60	滿	渝	游	溂	溪	溘	滉	溷	滓	溽	溯	滄	溲	滔	滕	溏
	5E70	溥	滂	溟	潁	漑	灌	滬	滸	滾	漿	滲	漱	滯	漲	滌	
	5F20		漾	漓	滷	澆	潺	潸	澁	澀	潯	潛	濳	潭	澂	潼	潘
	5F30	澎	澑	濂	潦	澳	澣	澡	澤	澹	濆	澪	濟	濕	濬	濔	濘
	5F40	濱	濮	濛	瀉	瀋	濺	瀑	瀁	瀏	濾	瀛	瀚	潴	瀝	瀘	瀟
	5F50	瀰	瀾	瀲	灑	灣											
火	5F50						炙	炒	炯	烱	炬	炸	炳	炮	烟	烋	烝
	5F60	烙	焉	烽	焜	焙	煥	煕	熈	煦	煢	煌	煖	煬	熏	燻	熄
	5F70	熕	熨	熬	燗	熹	熾	燒	燉	燔	燎	燠	燬	燧	燵	燼	
	6020		燹	燿	爍	爐	爛	爨									
爪	6020								爭	爬	爰	爲					
爻	6020												爻	爼			
爿	6020														爿	牀	牆
	6030	牋	牘														
牛	6030			牴	牾	犂	犁	犇	犒	犖	犢	犧					
犬	6030												犹	犲	狃	狆	狄
	6040	狎	狒	狢	狠	狡	狹	狷	倏	猗	猊	猜	猖	猝	猴	猯	猩
	6050	猥	猾	獎	獏	默	獗	獪	獨	獰	獸	獵	獻	獺			
玉	6050														珈	玳	珎
	6060	玻	珀	珥	珮	珞	璢	琅	瑯	琥	珸	琲	琺	瑕	琿	瑟	瑙
	6070	瑁	瑜	瑩	瑰	瑣	瑪	瑶	瑾	璋	璞	璧	瓊	瓏	瓔	珱	
瓜	6120		瓠	瓣													
瓦	6120				瓧	瓩	瓮	瓲	瓰	瓱	瓸	瓷	甄	甃	甅	甌	甎
	6130	甍	甕	甓													
甘	6130				甞												
生	6130					甦											
用	6130						甬										
田	6130							甼	畄	畍	畊	畉	畛	畆	畚	畩	畤
	6140	畧	畫	畭	畸	當	疆	疇	畴	疊	疉	疂					
病	6140												疔	疚	疝	疥	疣
	6150	痂	疳	痃	疵	疽	疸	疼	疱	痍	痊	痒	痙	痣	痞	痾	痿
	6160	痼	瘁	痰	痺	痲	痳	瘋	瘍	瘉	瘟	瘧	瘠	瘡	瘢	瘤	瘴
	6170	瘰	瘻	癇	癈	癆	癜	癘	癡	癢	癨	癩	癪	癧	癬	癰	
	6220		癲														
癶	6220			癶	癸	發											

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
白	6220						皀	皃	皈	皋	皎	皖	皓	皙	皚		
皮	6220															皰	皴
	6230	皸	皹	皺													
皿	6230				盂	盍	盖	盒	盞	盡	盥	盧	盪	蘯			
目	6230														盻	眈	眇
	6240	眄	眩	眤	眞	眥	眦	眛	眷	眸	睇	睚	睨	睫	睛	睥	睿
	6250	睾	睹	瞎	瞋	瞑	瞠	瞞	瞰	瞶	瞹	瞿	瞼	瞽	瞻	矇	矍
	6260	矗	矚														
矛	6260			矜													
矢	6260				矣	矮											
石	6260						矼	砌	砒	礦	砠	礪	硅	碎	硴	碆	硼
	6270	碚	碌	碣	碵	碪	碯	磑	磆	磋	磔	碾	碼	磅	磊	磬	
	6320		磧	磚	磽	磴	礇	礒	礑	礙	礬	礫					
示	6320												祀	祠	祗	祟	祚
	6330	祕	祓	祺	祿	禊	禝	禧	齋	禪	禮	禳					
禸	6330												禹	禺			
禾	6330														秉	秕	秧
	6340	秬	秡	秣	稈	稍	稘	稙	稠	稟	禀	稱	稻	稾	稷	穃	穗
	6350	穉	穡	穢	穩	龝	穰										
穴	6350							穹	穽	窈	窗	窕	窘	窖	窩	竈	窰
	6360	窶	竅	竄	窿	邃	竇	竊									
立	6360								竍	竏	竕	竓	站	竚	竝	竡	竢
	6370	竦	竭	竰													
竹	6370				笂	笏	笊	笆	笳	笘	笙	笞	笵	笨	笶	筐	
	6420		筺	笄	筍	笋	筌	筅	筵	筥	筴	筧	筰	筱	筬	筮	箝
	6430	箘	箟	箍	箜	箚	箋	箒	箏	筝	箙	篋	篁	篌	篏	箴	篆
	6440	篝	篩	簑	簔	篦	篥	籠	簀	簇	簓	篳	篷	簗	簍	篶	簣
	6450	簧	簪	簟	簷	簫	簽	籌	籃	籔	籏	籀	籐	籘	籟	籤	籖
	6460	籥	籬														
米	6460			籵	粃	粐	粤	粭	粢	粫	粡	粨	粳	粲	粱	粮	粹
	6470	粽	糀	糅	糂	糘	糒	糜	糢	鬻	糯	糲	糴	糶			
糸	6470														糺	紆	
	6520		紂	紜	紕	紊	絅	絋	紮	紲	紿	紵	絆	絳	絖	絎	絲
	6530	絨	絮	絏	絣	經	綉	絛	綏	絽	綛	綺	綮	綣	綵	緇	綽
	6540	綫	總	綢	綯	緜	綸	綟	綰	緘	緝	緤	緞	緻	緲	緡	縅
	6550	縊	縣	縡	縒	縱	縟	縉	縋	縢	繆	繦	縻	縵	縹	繃	縷
	6560	縲	縺	繧	繝	繖	繞	繙	繚	繹	繪	繩	繼	繻	纃	緕	繽
	6570	辮	繿	纈	纉	續	纒	纐	纓	纔	纖	纎	纛	纜			
缶	6570														缸	缺	
	6620		罅	罌	罍	罎	罐										

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
网	6620							网	罕	罔	罘	罟	罠	罨	罩	罧	罸
	6630	羂	羆	羃	羈	羇											
羊	6630						羌	羔	羞	羝	羚	羣	羯	羲	羹	羮	羶
	6640	羸	譱														
羽	6640			翅	翆	翊	翕	翔	翡	翦	翩	翳	翹	飜			
老	6640														耆	耄	耋
耒	6650	耒	耘	耙	耜	耡	耨										
耳	6650							耿	耻	聊	聆	聒	聘	聚	聟	聢	聨
	6660	聳	聲	聰	聶	聹	聽										
聿	6660							聿	肄	肆	肅						
肉	6660											肛	肓	肚	肭	冐	肬
	6670	胛	胥	胙	胝	胄	胚	胖	脉	胯	胱	脛	脩	脣	脯	腋	
	6720		隋	腆	脾	腓	腑	胼	腱	腮	腥	腦	腴	膃	膈	膊	膀
	6730	膂	膠	膕	膤	膣	腟	膓	膩	膰	膵	膾	膸	膽	臀	臂	膺
	6740	臉	臍	臑	臙	臘	臈	臚	臟	臠							
臣	6740										臧						
至	6740											臺	臻				
臼	6740													臾	舁	舂	舅
	6750	與	舊														
舌	6750			舍	舐	舖											
舟	6750						舩	舫	舸	舳	艀	艙	艘	艝	艚	艟	艤
	6760	艢	艨	艪	艫	舮											
艮	6760						艱										
色	6760							艷									
艸	6760								艸	艾	芍	芒	芫	芟	芻	芬	苡
	6770	苣	苟	苒	苴	苳	苺	莓	范	苻	苹	苞	茆	苜	茉	苙	
	6820		茵	茴	茖	茲	茱	荀	茹	荐	荅	茯	茫	茗	茘	莅	莚
	6830	莪	莟	莢	莖	茣	莎	莇	莊	荼	莵	荳	荵	莠	莉	莨	菴
	6840	萓	菫	菎	菽	萃	菘	萋	菁	菷	萇	菠	菲	萍	萢	萠	莽
	6850	萸	蔆	菻	葭	萪	萼	蕚	蒄	葷	葫	蒭	葮	蒂	葩	葆	萬
	6860	葯	葹	萵	蓊	葢	蒹	蒿	蒟	蓙	蓍	蒻	蓚	蓐	蓁	蓆	蓖
	6870	蒡	蔡	蓿	蓴	蔗	蔘	蔬	蔟	蔕	蔔	蓼	蕀	蕣	蕘	蕈	
	6920		蕁	蘂	蕋	蕕	薀	薤	薈	薑	薊	薨	蕭	薔	薛	藪	薇
	6930	薜	蕷	蕾	薐	藉	薺	藏	薹	藐	藕	藝	藥	藜	藹	蘊	蘓
	6940	蘋	藾	藺	蘆	蘢	蘚	蘰	蘿								
虍	6940									虍	乕	虔	號	虧			

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
虫	6940														虱	蚓	蚣
	6950	蚩	蚪	蚋	蚌	蚶	蚯	蛄	蛆	蚰	蛉	蠣	蚫	蛔	蛞	蛩	蛬
	6960	蛟	蛛	蛯	蜒	蜆	蜈	蜀	蜃	蛻	蜑	蜉	蜍	蛹	蜊	蜴	蜿
	6970	蜷	蜻	蜥	蜩	蜚	蝠	蝟	蝸	蝌	蝎	蝴	蝗	蝨	蝮	蝙	
	6A20		蝓	蝣	蝪	蠅	螢	螟	螂	螯	蟋	螽	蟀	蟐	雖	螫	蟄
	6A30	螳	蟇	蟆	螻	蟯	蟲	蟠	蠏	蠍	蟾	蟶	蟷	蠎	蟒	蠑	蠖
	6A40	蠕	蠢	蠡	蠱	蠶	蠹	蠧	蠻								
血	6A40									衄	衂						
行	6A40											衒	衙	衞	衢		
衣	6A40															衫	袁
	6A50	衾	袞	衵	衽	袵	衲	袂	袗	袒	袮	袙	袢	袍	袤	袰	袿
	6A60	袱	裃	裄	裔	裘	裙	裝	裹	褂	裼	裴	裨	裲	褄	褌	褊
	6A70	褓	襃	褞	褥	褪	褫	襁	襄	褻	褶	褸	襌	褝	襠	襞	
	6B20		襦	襤	襭	襪	襯	襴	襷								
襾	6B20									覀	覃	覈	覊				
見	6B20													覓	覘	覡	覩
	6B30	覦	覬	覯	覲	覺	覽	覿	觀								
角	6B30									觚	觜	觝	觧	觴	觸		
言	6B30															訃	訖
	6B40	訐	訌	訛	訝	訥	訶	詁	詛	詒	詆	詈	詼	詭	詬	詢	誅
	6B50	誂	誄	誨	誡	誑	誥	誦	誚	誣	諄	諍	諂	諚	諫	諳	諧
	6B60	諤	諱	謔	諠	諢	諷	諞	諛	謌	謇	謚	諡	謖	謐	謗	謠
	6B70	謳	鞫	謦	謫	謾	謨	譁	譌	譏	譎	證	譖	譛	譚	譫	
	6C20		譟	譬	譯	譴	譽	讀	讌	讎	讒	讓	讖	讙	讚		
谷	6C20															谺	豁
	6C30	谿															
豆	6C30		豈	豌	豎	豐											
豕	6C30						豕	豢	豬								
豸	6C30									豸	豺	貂	貉	貅	貊	貍	貎
	6C40	貔	豼	貘													
貝	6C40				戝	貭	貪	貽	貲	貳	貮	貶	賈	賁	賤	賣	賚
	6C50	賽	賺	賻	贄	贅	贊	贇	贏	贍	贐	齎	贓	賍	贔	贖	
赤	6C50																赧
	6C60	赭															
走	6C60		赱	赳	趁	趙											
足	6C60						跂	趾	趺	跏	跚	跖	跌	跛	跋	跪	跫
	6C70	跟	跣	跼	踈	踉	跿	踝	踞	踐	踟	蹂	踵	踰	踴	蹊	
	6D20		蹇	蹉	蹌	蹐	蹈	蹙	蹤	蹠	踪	蹣	蹕	蹶	蹲	蹼	躁
	6D30	躇	躅	躄	躋	躊	躓	躑	躔	躙	躪	躡					
身	6D30												躬	躰	軆	躱	躾
	6D40	軅	軈														

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
車	6D40			軋	軛	軣	軼	軻	軫	軾	輊	輅	輕	輒	輙	輓	輜
	6D50	輟	輛	輌	輦	輳	輻	輹	轅	轂	輾	轌	轉	轆	轎	轗	轜
	6D60	轢	轣	轤													
辛	6D60				辜	辟	辣	辭	辯								
辶	6D60									辷	迚	迥	迢	迪	迯	邇	迴
	6D70	逅	迹	迺	逑	逕	逡	逍	逞	逖	逋	逧	逶	逵	逹	迸	
	6E20		遏	遐	遑	遒	逎	遉	逾	遖	遘	遞	遨	遯	遶	隨	遲
	6E30	邂	遽	邁	邀	邊	邉	邏									
邑	6E30								邨	邯	邱	邵	郢	郤	扈	郛	鄂
	6E40	鄒	鄙	鄲	鄰												
酉	6E40					酊	酖	酘	酣	酥	酩	酳	酲	醋	醉	醂	醢
	6E50	醫	醯	醪	醵	醴	醺	釀	釁								
釆	6E50									釉	釋						
里	6E50											釐					
金	6E50												釖	釟	釡	釛	釼
	6E60	釵	釶	鈞	釿	鈔	鈬	鈕	鈑	鉞	鉗	鉅	鉉	鉤	鉈	銕	鈿
	6E70	鉋	鉐	銜	銖	銓	銛	鉚	鋏	銹	銷	鋩	錏	鋺	鍄	錮	
	6F20		錙	錢	錚	錣	錺	錵	錻	鍜	鍠	鍼	鍮	鍖	鎰	鎬	鎭
	6F30	鎔	鎹	鏖	鏗	鏨	鏥	鏘	鏃	鏝	鏐	鏈	鏤	鐚	鐔	鐓	鐃
	6F40	鐇	鐐	鐶	鐫	鐵	鐡	鐺	鑁	鑒	鑄	鑛	鑠	鑢	鑞	鑪	鈩
	6F50	鑰	鑵	鑷	鑽	鑚	鑼	鑾	钁	鑿							
門	6F50										閂	閇	閊	閔	閖	閘	閙
	6F60	閠	閨	閧	閭	閼	閻	閹	閾	闊	濶	闃	闍	闌	闕	闔	闖
	6F70	關	闡	闥	闢												
阜	6F70					阡	阨	阮	阯	陂	陌	陏	陋	陷	陜	陞	
	7020		陝	陟	陦	陲	陬	隍	隘	隕	隗	險	隧	隱	隲	隰	隴
隶	7030	隶	隸														
隹	7030			隹	雎	雋	雉	雍	襍	雜	霍	雕					
雨	7030												雹	霄	霆	霈	霓
	7040	霎	霑	霏	霖	霙	霤	霪	霰	霹	霽	霾	靄	靆	靈	靂	靉
青	7050	靜															
非	7050		靠														
面	7050			靤	靦	靨											
革	7050						勒	靫	靱	靹	鞅	靼	鞁	靺	鞆	鞋	鞏
	7060	鞐	鞜	鞨	鞦	鞣	鞳	鞴	韃	韆	韈						
韋	7060											韋	韜				
韭	7060													韭	齏	韲	
音	7060																竟
	7070	韶	韵														
頁	7070			頏	頌	頸	頤	頡	頷	頽	顆	顏	顋	顫	顯	顰	
	7120		顱	顴	顳												

	16進コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
風	7120					颪	颯	颱	颶	飄	飃	飆					
食	7120												飩	飫	餃	餉	餒
	7130	餔	餘	餡	餝	餞	餤	餠	餬	餮	餽	餾	饂	饉	饅	饐	饋
	7140	饑	饒	饌	饕												
首	7140					馗	馘										
香	7140							馥									
馬	7140								馭	馮	馼	駟	駛	駝	駘	駑	駭
	7150	駮	駱	駲	駻	駸	騁	騏	騅	駢	騙	騫	騷	驅	驂	驀	驃
	7160	騾	驕	驍	驛	驗	驟	驢	驥	驤	驩	驫	驪				
骨	7160													骭	骰	骼	髀
	7170	髏	髑	髓	體												
高	7170					髞											
髟	7170						髟	髢	髣	髦	髯	髫	髮	髴	髱	髷	
	7220		髻	鬆	鬘	鬚	鬟	鬢	鬣								
鬥	7220									鬥	鬧	鬨	鬩	鬪	鬮		
鬯	7220															鬯	
鬲	7220																鬲
鬼	7230	魄	魃	魏	魍	魎	魑	魘									
魚	7230								魴	鮓	鮃	鮑	鮖	鮗	鮟	鮠	鮨
	7240	鮴	鯀	鯊	鮹	鯆	鯏	鯑	鯒	鯣	鯢	鯤	鯔	鯡	鰺	鯲	鯱
	7250	鯰	鰕	鰔	鰉	鰓	鰌	鰆	鰈	鰒	鰊	鰄	鰮	鰛	鰥	鰤	鰡
	7260	鰰	鱇	鰲	鱆	鰾	鱚	鱠	鱧	鱶	鱸						
鳥	7260											鳧	鳬	鳰	鴉	鴈	鳫
	7270	鴃	鴆	鴪	鴦	鶯	鴣	鴟	鵄	鴕	鴒	鵁	鴿	鴾	鵆	鵈	
	7320		鵝	鵞	鵤	鵑	鵐	鵙	鵲	鶉	鶇	鶫	鵯	鵺	鶚	鶤	鶩
	7330	鶲	鷄	鷁	鶻	鶸	鶺	鷆	鷏	鷂	鷙	鷓	鷸	鷦	鷭	鷯	鷽
	7340	鸚	鸛	鸞													
鹵	7340				鹵	鹹	鹽										
鹿	7340							麁	麈	麋	麌	麒	麕	麑	麝		
麦	7340															麥	麩
	7350	麸	麪	麭													
麻	7350				靡												
黄	7350					黌											
黍	7350						黎	黏	黐								
黒	7350									黔	黜	點	黝	黠	黥	黨	黯
	7360	黴	黶	黷													
黹	7360				黹	黻	黼										
黽	7360							黽	鼇	鼈							
皷	7360										皷	鼕					
鼠	7360												鼡	鼬			
鼻	7360														鼾		

	16進コード	0	1		3	4	5	6	7	8	9	A	B	C	D	E	F
齊	7360															齊	
齒	7360																齒
	7370	齔	齣	[illegible]			[illegible]	齧	齬	齪	齷	齲	齶				
龍	7370													龕			
龜	7370														龜		
龠	7370															龠	

ローマ字変換

MSX$_{2+}$-BASICではひらがなやカタカナをキーボードからローマ字で入力し，かなに変換することができます。SHIFTキーを押しながらかなキーを押してください。変換は下の表に従って行われます。

アルファベットの小文字はひらがなに，大文字はカタカナに変換されます。

日本語入力モードについては，第1章「日本語の入力」を参照してください。

	A	I	U	E	O	YA	YI	YU	YE	YO	HA	HI	HU	HE	HO	SU	N
	あ	い	う	え	お												
B	ば	び	ぶ	べ	ぼ	びゃ	びぃ	びゅ	びぇ	びょ							
C						ちゃ	ちぃ	ちゅ	ちぇ	ちょ	ちゃ	ち	ちゅ	ちぇ	ちょ		
D	だ	ぢ	づ	で	ど	ぢゃ	ぢぃ	ぢゅ	ぢぇ	ぢょ	でゃ	でぃ	でゅ	でぇ	でょ		
F	ふぁ	ふぃ	ふ	ふぇ	ふぉ	ふゃ	ふぃ	ふゅ	ふぇ	ふょ							
G	が	ぎ	ぐ	げ	ご	ぎゃ	ぎぃ	ぎゅ	ぎぇ	ぎょ							
H	は	ひ	ふ	へ	ほ	ひゃ	ひぃ	ひゅ	ひぇ	ひょ							
J	じゃ	じ	じゅ	じぇ	じょ	じゃ	じぃ	じゅ	じぇ	じょ							
K	か	き	く	け	こ	きゃ	きぃ	きゅ	きぇ	きょ							
L	ぁ	ぃ	ぅ	ぇ	ぉ	ゃ	ぃ	ゅ	ぇ	ょ							
M	ま	み	む	め	も	みゃ	みぃ	みゅ	みぇ	みょ							
N	な	に	ぬ	ね	の	にゃ	にぃ	にゅ	にぇ	にょ							ん
P	ぱ	ぴ	ぷ	ぺ	ぽ	ぴゃ	ぴぃ	ぴゅ	ぴぇ	ぴょ							
R	ら	り	る	れ	ろ	りゃ	りぃ	りゅ	りぇ	りょ							
S	さ	し	す	せ	そ	しゃ	しぃ	しゅ	しぇ	しょ	しゃ	し	しゅ	しぇ	しょ		
T	た	ち	つ	て	と	ちゃ	ちぃ	ちゅ	ちぇ	ちょ	てゃ	てぃ	てゅ	てぇ	てょ	つ	
W	わ	うぃ	う	うぇ	を												
Y	や	い	ゆ	いぇ	よ												
Z	ざ	じ	ず	ぜ	ぞ	じゃ	じぃ	じゅ	じぇ	じょ							
V	ゔぁ	ゔぃ	ゔ	ゔぇ	ゔぉ												

・「ん」Nの後にY以外の子音を入れる。

・「ッ」N以外の同じ子音が2つ続く。

ローマ字カナ変換モードにすると，次の文字はカナ記号に変換されます。

”X”	→	「−」	(長音)
”*”	→	「—」	(長音)
”:”	→	「〃」	(濁点)
”]”	→	「。」	(句点)
”_”	→	「・」	(中点)
”?”	→	「°」	(半濁点)
”>”	→	「、」	(読点)
” {”	→	「」」	(右かっこ)
”{ ”	→	「「」	(左かっこ)

上記以外のキーは英字モードのときと同じ働きをします。

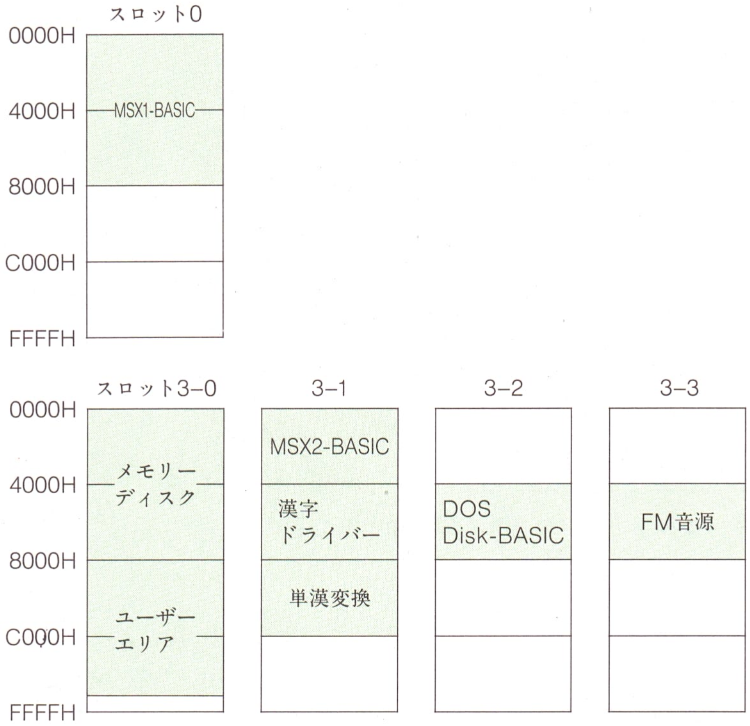
キーコード表

MSX$_{2+}$-BASICではエディットキーやその他の制御キーを押すと，次のようなコードが得られます。

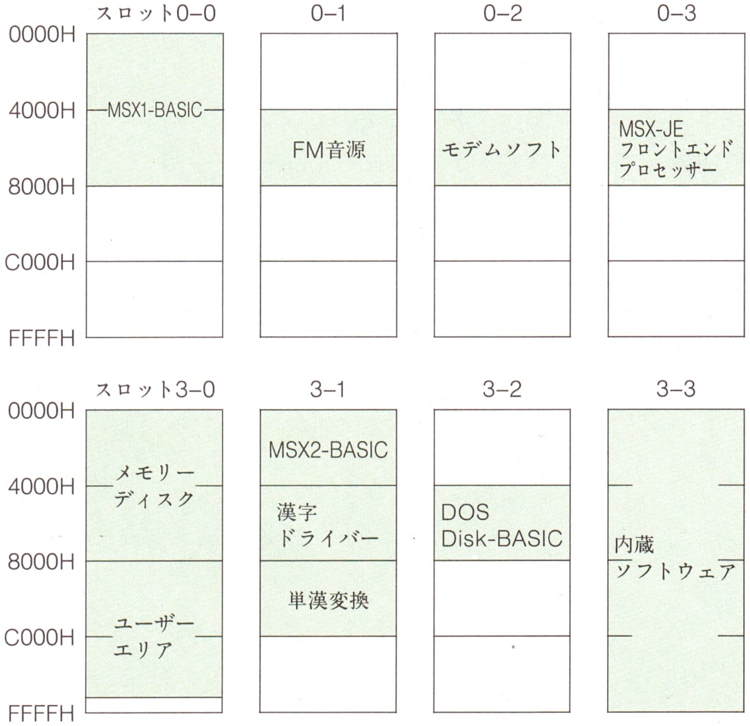
コード	キー	コード	キー
8	BS	27	ESC
9	TAB	28	→ cursor right
11	HOME	29	← cursor left
12	CLS	30	↑ cursor up
13	RETURN	31	↓ cursor down
18	INS	127	DEL
24	SELECT		

メモリーマップ

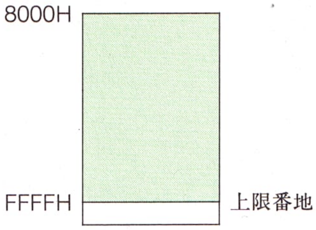
MSX2→MSX2+の拡張にスロット３のみを使用する場合



MSX2→MSX2+の拡張にスロット0とスロット3を使用する場合



ユーザーエリアの上限番地



ユーザーエリアの上限番地は、BASICスタート後、何もしない状態のときに以下のコマンドを実行することにより調べることができます。

```
J=&HFC4A:PRINT HEX$(PEEK(J+1)*256+PEEK(J))
```

FC4AHに下位、FC4BHに上位のアドレスが入っています。

ユーザーエリアの構成

プログラム領域
変数領域
配列変数領域
フリーエリア
スタックエリア
文字列領域
ファイルコントロール ブロック

プログラム領域

行番号をつけたプログラムが格納される。

変数領域

数値型データおよび，文字型データへのポインターが格納される。

配列変数領域

配列変数のデータが格納される。文字型の場合，文字列領域へのポインターが格納される。

フリーエリア

未使用領域。FRE関数で大きさを求めることができる。

スタックエリア

戻りアドレスを退避させておくためのスタック領域。

文字列領域

文字型の変数や配列変数に含まれる文字列が格納される。
CLEAR文で大きさを指定できる。

ファイルコントロールブロック

ファイルの入出力時に使用される。

I/Oポートの割り付け

用　　途	ポート番号 (16進)	内　　容
FM音源LSI (MSX-MUSIC)	7C 7D	アドレスレジスター データレジスター
VDP for MSX 1 adapter	88 89 8A 8B	データ読み出し/書き込み コマンド・アドレス書き込み ステータスレジスター読み出し パレットレジスター書き込み レジスター間接書き込み
モデム	8C 8D	—— ——
プリンター	90 91	ライト時　bit 0：ストローブ リード時　bit 1：ステータス プリンターデータ
VDP	98 99 9A 9B	88に同じ 89に同じ 8Aに同じ 8Bに同じ
PSG	A0 A1 A2	アドレスラッチ（書き込み） データ書き込み データ読み出し
PPI	A8 A9 A8 A9	ポートA(メモリースロットセレクト)用のデータ読み出し/書き込み ポートB(キーボードスキャン)用のデータ読み出し/書き込み ポートC（カセット）用のデータ読み出し/書き込み モードセット（書き込み）
MSX-ENGINE	AC～AF	——
拡張メモリー	B0 B1 B2 B3	アドレスA0～A7 アドレスA8～A10，A13～A15 コントロール アドレスA11～A12，D0～D 7 モードセット
カレンダークロック	B4 B5	アドレスラッチ データ

ライトペン	B8～BA BB	リード／ライト ライト　オンリー
VHDコントロール	BC BD BE BF	ポートA ポートB ポートC モードセット
MSX-Audio	C0～C1	——
漢字ROM	D8 D9	b5～b0　下位アドレス　ライトオンリー b5～b0　上位アドレス　ライト時 b7～b0　データ　リード時
システムコントロール	F5	b0　漢字ROM b1　漢字のために予約 b2　MSX-Audio b3　スーパーインポーズ b4　MSX-interface b5　RS-232C b6　ライトペン b7　カレンダークロック（for MSX 1）
カラーバス I/O	F6	——
AVコントロール	F7	b0　Audio R　0：Mixing on b1　Audio L　0：Mixing on b2　Video input select　0：21 pin b3　Video input sence　0：No input b4　AV control　0：TV b5　Ym control　0：TV b6　Ys control　0：Super b7　Video select　0：TV
メモリーマッパー	FC～FF	——

MSX-DOSについて

MSX-DOSの概要

MSX-DOSは、MSXコンピューターのためのDOS(ディスクオペレーティングシステム)です。

オペレーティングシステムとは、機械(ハードウェア)に一番近い部分のプログラム(ソフトウェア)で、ビジネスソフトや、ゲームなどのほとんどのアプリケーションプログラムは、オペレーティングシステムの助けをかりて動いているのです。コンピューターの言語も、オペレーティングシステムのもとで使われます。

DOSもオペレーティングシステムの一種で、図書館の本を管理している司書に例えられます。ある本が読みたいと思ったときに、その本を探して貸してくれるように、DOSは、ディスク内のファイルを管理し、求められたファイルを見つけ出してくれるのです。

MSX-DOSを使うには、2つのファイルが必要です。

COMMAND. COM	MSX-DOSのコマンドを実行する
MSXDOS. SYS	MSX-DOSの本体

MSX-DOSがロードされると、メモリーマップは図のようになります。

●MSX-DOSを動かすには、64Kバイト以上のRAMが必要です。

MSX-DOSの起動

1 システムディスクをドライブAに入れます。

2 外部ドライブを使うときは、ドライブの電源を入れます。

3 コンピューターとモニターの電源を入れます。

MSX (version 1.0)のコンピューターの場合

```
MSX-DOS version x.xx
Copyright 1984 by Microsoft

COMMAND version x.xx

Current date is xxx     xx-xx-xx
Enter new date:_
```

というメッセージが出ます。年－月－日という形式で日付けを入力するか、

RETURN キーをただ押してください。

MSX2およびMSX$_{2+}$のコンピューターの場合は，このメッセージは表示されません。

次に

```
MSX-DOS version x.xx
Copyright 1984 by Microsoft

COMMAND version x.xx
```

というメッセージが表示され、A>というサインがあらわれます。これはMSX-DOSのプロンプトと呼ばれ、MSX-DOSが現在ドライブAを使っているということと、MSX-DOSが起動されてコマンドを待っているということを表わしています。

MSX-DOSシステムのバックアップ

まちがってファイルを消してしまったり、ディスクが使えなくなったときのために、システムディスクのバックアップコピー(予備)を作っておくことをおすすめします。

1 MSX-DOSを起動させます。

2 コピーに使う空のディスクをフォーマットします。「MSX-DOSのコマンド」のFORMATの項をご覧ください。

3 MSXDOS. SYSをコピーします。

2台以上のドライブを使えるときは、システムディスクをドライブAに、フォーマットした空のディスクをドライブBに入れて、

COPY　A：MSXDOS. SYS　B：　RETURN

と打ちこんでください。

ドライブが1台しかないときは、はじめに

COPY　A：MSXDOS. SYS　B：　RETURN

と打ちます。すると

```
Insert diskette for drive B:
and strike a key when ready
```

というメッセージがあらわれます。システムディスクをドライブから取り出し、新しいディスクを入れて何かキーを押すとコピーが始まります。

コピーが終わると、次のメッセージが表示されます。

```
1 File copied
A>
```

4 同様に、

COPY　A：COMMAND. COM　B：　RETURN

で、COMMAND. COMもコピーします。

● COPY　A：＊.＊　B：　RETURN

と入力すると、ディスクの内容を全部1度にコピーすることもできますが、ディスクにたくさんのファイルがあると、メッセージに従って何回か2枚のディスクを入れかえ、どれかひとつキーを押すという作業をくり返さなければならずコピーに時間がかかります。

ディスクエラー

コマンドやプログラムの実行中にディスクに関するエラーが起こると、次のメッセージが表示されます。

```
xxx error xxxxx drive x
Abort, Retry, Ignore?
```

それぞれの頭文字、AかRかIを押してください。次のようになります。

A：プログラムの実行を中止し、プロンプトが出ます。

R：もう一度プログラムを実行します。

I：エラーの発生した命令を無視して先へ進みます。通常は使わないでください。

MSX-DOSの終了

1 ドライブのIN USE ランプが消えているか確かめて、ドライブにはいっているディスクをすべて取り出します。

2 コンピューターやドライブ、その他の機器の電源を切ります。

MSX-DOSのコマンド

MSX-DOSには、13のコマンドがあります。この章では各コマンドをアルファベット順に説明します。コマンドの形式は次のとおりです。

デリート
DEL (delete) ← コマンド名

指定されたディスク上のファイルを消去する。 ← 機能

形　式

DEL␣[ドライブ名]ファイル名.タイプ名

- [　]は、中の項目を省略できることを示しています。
- タイプ名：ファイルの種類
- ファイル名：ファイルの名前。ファイル名とピリオド(.)とタイプ名でファイルを指定します。
- [ドライブ名]：ディスクドライブの指定.(後ろにコロン(:)をつけます。)
- ␣：コマンドの後には、スペースを区切り記号としていれてください。ここでは、␣という記号で表わしています。

ドライブ名	条	A:、B:、C:、D:、E:、F:、G:、H:	← 入力の条件
	省	現在使用中のディスクドライブ	← 入力を省略したとき
ファイル名	条	8字以内の文字列。	
タイプ名	条	3字以内の文字列。	

働きと使いかた

実行例

← コマンドについての補足説明や、そのコマンドを使った実行例。

●MSX-DOSでは、BASICとちがってタイプ名は原則として省略できません。ただし、BASICのファイルで、タイプ名がないものについては、省略することができます。

各コマンドの説明を読むときに、知っていないと困る言葉が3つあります。

ヌルストリング

文字もスペースも何も含まない文字列。BASICのファイルでタイプ名を特に指定しなかった場合、ヌルストリングが自動的に指定されます。

バッチファイル

MSX-DOSでは、BASICのように行番号をつけてプログラムを簡単につくることはできません。MSX-DOS上で使うコンピューター言語や、編集のためのプログラムが必要です。けれども、MSX-DOSのコマンドや、すでにできあがっているプログラムファイルの名前を、実行する順番に並べたバッチファイルを作っておけば、他のMSX-DOSのコマンドのようにファイル名を入力するだけで実行することができます。

バッチファイルを作るには、

COPY CON ファイル名. BAT [RETURN]

(バッチファイルには、必ず.BATというタイプ名をつけます。)を実行して、1行ずつコマンドやファイル名を入力し、最後に [CTRL] キーを押しながら [Z] キーを押して、[RETURN] キーを押します。

AUTOEXEC. BATというバッチファイルを作っておくと、MSX-DOSを起動するたびに自動的にそのプログラムが実行されます。

ワイルドカード

ファイル名やタイプ名のかわりに使えるジョーカーのようなものです。? (クエスチョン・マーク)はすべての文字(1文字)のかわりに、*(アステリスク)は、すべての文字列(何文字でも可)のかわりになります。

ディスクドライブを１台だけ使っている場合

ドライブが１台しかない場合でも、コマンドを使うときは、ドライブＡとドライブＢがあるつもりになってください。２枚のディスクを入れ換えることによって、１台のディスクを２台分に使うことができます。

ドライブＡのディスクを使った後、ドライブＢを指定すると、

```
Insert diskette for drive B:
and strike a key when ready
```

というメッセージが表示されます。ドライブからディスクを抜いて、別のドライブＢ用のディスクを入れてください。

その次にまたドライブＡを指定すると、

```
Insert diskette for drive A:
and strike a key when ready
```

というメッセージが表示されます。ドライブＢ用のディスクを取り出して、もとのディスクを入れればよいわけです。

BASIC（ベーシック）

MSX-DOSからMSX-Disk BASICを起動し、指定したプログラムをロードして実行する。

形　式

BASIC␣[[ドライブ名]ファイル名[.タイプ名]]

ドライブ名	条	A：、B：、C：、D：、E：、F：、G：、H：
	省	現在使用中のディスクドライブ。
ファイル名	条	8字以内の文字列。
	省	Disk BASIC起動のみ。
タイプ名	条	3字以内の文字列。
	条	ヌルストリング。

働きと使いかた

BASICのプログラムをファイル名[.タイプ名]で指定すると、BASIC起動後そのプログラムがロードされ、実行されます。

- このコマンドは、コンピューター内部のスロットを切り換えて、ROMに入っているDisk BASICを起動します。したがって、MSX-DOS起動時とMSX-Disk BASIC起動時では異なったメモリーマップになっています。
- MSX-Disk BASICからMSX-DOSに戻るときは、CALL SYSTEM文を使います。(CALL SYSTEM文は、最初にMSX-DOSを起動してBASICコマンドでDisk BASICに移ったときにしか使えません。)

実行例

```
BASIC B:PROG4.BAS
```

MSX-Disk BASICを起動して、ドライブBにあるディスクのPROG4. BASというプログラムをロードし、実行します。

COPY (copy)

コピー

ディスク上のファイルを同一ディスクまたは他のディスク上にコピーする。

形　式

COPY␣[ドライブ名1]ファイル名1.タイプ名1[+……]␣[ドライブ名2][ファイル名2.タイプ名2]

ドライブ名　　条 A:、B:、C:、D:、E:、F:、G:、H:
　　　　　　　省 現在使用中のディスクドライブ。
ファイル名　　条 8字以内の文字列。
タイプ名　　　条 3字以内の文字列。

働きと使いかた

ドライブ名1のディスクからドライブ名2のディスクへ内容をファイル単位でコピーします。

ファイル名とタイプ名には、ワイルドカードが使えます。ドライブ名2を省略すると、そのときに使っているドライブ上のディスクにコピーされます。ファイル名2とタイプ名2を省略すると、コピーは元のファイルと同じ名前になります。けれども、同一ドライブ上に同じ名前でコピーすることはできません。

いくつかのファイル名1.タイプ名1を+記号でつないで入力すると、それらが合併されたファイルがつくられます。

実行例

```
COPY ABC.DAT XYZ.PRS
```

を実行すると、そのとき使っているドライブにあるABC.DATというファイルが、同じディスクにXYZ.PRSという名前でコピーされています。

```
COPY A:ABC.C+BCD.C B:ABCD.C
```

では、ドライブAにあるABC.CとBCD.Cが合併されたファイルがドライブBのディスクにABCD.Cという名前で作られます。

```
COPY A:*.COM B:
```

では、ドライブAにある.COMというタイプ名のファイルがすべてドライブBのディスクに、同じ名前でコピーされます。

DATE (date)

日付けを表示し、変更する。

形　式

DATE␣[年－月－日]

年－月－日　条 年：80～99（19を省略）、00～79(20を省略)
または 1980～2079
月：1～12
日：1～31
それぞれの間はハイフン(－)かスラッシュ(/)で区切る。

働きと使いかた

DATE RETURN を実行すると、次のメッセージを表示し、変更を求めます。

Current date is　曜日　年－月－日
Enter new date：

メッセージに表示された日付けを変更しなくてよいときは、RETURN キーを押します。

MSX-DOSは月の大、小やうるう年も正しく判別できるようになっています。

● 入力が条件に合わない場合

```
Invalid date
Enter new date:
```

のメッセージが、正しい入力が行われるまで、繰り返し表示されます。

実行例

```
DATE 85-9-25
```

DEL (delete)

デリート

指定されたディスク上のファイルを消去する。

形　式

DEL␣[ドライブ名]ファイル名.タイプ名

ドライブ名　条 A：、B：、C：、D：、E：、F：、G：、H：

　　　　　　省 現在使用中のディスクドライブ。

ファイル名　条 8字以内の文字列。

タイプ名　　条 3字以内の文字列。

働きと使いかた

ERASEコマンドと同じ働きで、指定されたファイルをすべて消去します。
ファイル名とタイプ名にはワイルドカードが使えます。

実行例

```
DEL A:TEST.DAT
```

ドライブAのTEST. DATというファイルを探し、あればそのファイルを消去します。

```
DEL A:TEST??.DAT
```

ここでは、ファイル名の最初の4文字がTESTである6文字までのファイル名で、タイプ名がDATであるすべてのドライブAのファイル(例えばTEST15. DAT)が消去されます。

```
DEL A:TEST.*
```

では、タイプ名が何であっても、TESTというファイル名を持つドライブAのディスクのファイル(例えばTEST. BAS)が、すべて消去されます。

```
DEL A:*.*
```

を実行すると、〝Are you sure？(Y/N)〟が表示されます。Y(Yes)を入力すると、ドライブAのディスク上のファイルはすべて消去されます。

DIR (directory)

ディレクトリー(ファイルに関する情報がしまってある所)内のファイル名を表示する。

形　式

DIR␣[ドライブ名][ファイル名.タイプ名] [/P] [/W]

ドライブ名　㊇ A:、B:、C:、D:、E:、F:、G:、H:

㊔ 現在使用中のディスクドライブ。

ファイル名　㊇ 8字以内の文字列。

タイプ名　㊇ 3字以内の文字列。

働きと使いかた

ファイル名とそのファイルの大きさ(バイト数)，最後に修正をうけた日付けが表示されます。時計のついているコンピューターで、1行に36文字以上表示している場合は、最後に修正をうけた時刻も表示されます。

ファイル名とタイプ名には、ワイルドカードが使えます。また、省略した場合も、*がはいっているとみなされます。

/P をつけた場合画面がいっぱいになるといったん表示が止まります。その状態でキーをどれかひとつ押すと残りが表示されます。

/W をつけた場合は、ファイル名だけを1行に詰められるだけ詰めて表示します。

実行例

```
DIR
```

で、現在使用中のドライブ上のディスクにあるファイルをすべて表示します。

```
DIR A:
```

では、ドライブA上のディスクにあるすべてのファイル名が、表示されます。

```
DIR A:TEST.*
```

では、ドライブA上でファイル名がTESTであるファイル名がすべて表示されます。

```
DIR A:TEST.DAT
```

では、ドライブAのディスクでTEST.DATというファイルを探し、あればその情報を表示します。

ERASE（erase）

（イレーズ）

指定されたディスク上のファイルを消去する。

形　式

ERASE␣[ドライブ名]ファイル名.タイプ名

ドライブ名　条 A：、B：、C：、D：、E：、F：、G：、H：

　　　　　　省 現在使用中のディスクドライブ。

ファイル名　条 8字以内の文字列。

タイプ名　　条 3字以内の文字列。

働きと使いかた

DELコマンドと同じで、指定されたファイルをすべて消去します。
ファイル名とタイプ名にはワイルドカードが使えます。

実行例

```
ERASE A:TEST.DAT
```

ドライブAのTEST.DATというファイルを探し、あればそのファイルを消去します。

```
ERASE A:TEST.*
```

では、ファイル名がTESTであるすべてのドライブAのファイルが消去されます。

```
ERASE A:*.*
```

を実行すると、〝Are you sure？(Y/N)〟が表示されます。Y(Yes)を入力すると、ドライブA上のファイルはすべて消されてしまいます。

FORMAT (format)

フォーマット

ディスクを、MSX-DOSおよびMSX-Disk BASICで使えるようにフォーマットする。

形　式

FORMAT

働きと使いかた

新しいディスクは、使う前にフォーマットしなければなりません。MSX-DOS のフォーマットと、MSX-Disk BASICのフォーマットは同じですから、フォーマットされたディスクはどちらにも使うことができます。操作手順も Disk BASIC の FORMAT コマンドとほとんど同じです。

FORMAT

を実行すると、

```
Drive name? (A,B)
```

が表示されます. ドライブが1台のときは A キーを、2台のときは A キーか B キーのどちらかを押してください。

ディスクドライブを片面用として使っているときは次の表示があらわれます。

```
Strike a key when ready
```

フォーマットしたいディスクをドライブA(指定したドライブ)に入れ、キーボード上のキーを何かひとつ押します。

ディスクドライブを両面用として使っているときは、

```
1 - Single sided, 9 sectors
2 - Double sided, 9,sectors
```

という表示があらわれます。片面のディスクをフォーマットするときは1を、両面のディスクのときは2を指定してください。

指定すると、次の表示があらわれます。

```
Strike a key when ready
```

ディスクをドライブに入れて、キーを何かひとつ押してください。

フォーマットが終了すると、

```
Format complete
```

と表示され、プロンプトが出ます。

● FORMAT コマンドを途中で止めるとき(ただし Strike a key when ready で何かキーを押したあとは止められません。)は、CTRL キーを押しながら STOP キーを押します。

MODE(mode)

(モード)

画面の1行に表示する最大の文字数を決める。

形　式

MODE␣字数

字数　条 1≦ 数 ≦40　(MSXのとき)
　　　　1≦ 数 ≦80　(MSX2およびMSX_{2+}のとき)

働きと使いかた

字数が1から32のときは、スクリーンモードは1になります。32から上のときは、スクリーンモードは0になります。

MSX-DOSを起動したときは，字数はMSXでは29，MSX2およびMSX_{2+}では80になっています。

実行例

```
MODE 35
```

PAUSE (pause)

ポーズ

バッチファイルの実行を一時停止する。

形　式

PAUSE␣[注釈文]

働きと使いかた

バッチファイルを実行中に、ディスクを交換するなどプログラムを一時止めなければならない場合があります。あらかじめバッチファイルの中にこのコマンドを入れておけば、実行がそこまできたとき、そこで止めておくことができます。このとき、

```
Strike a key when ready
```

が表示されます。

ここで CTRL キーを押しながら C キーを押すと、

```
Terminate batch file (Y/N)?
```

と表示され、Y キーを押すと、バッチファイルの実行が中止されてMSX-DOSのプロンプトが出ます。

CTRL + C 以外のキーを押すか、Terminate batch file (Y/N) ? で N キーを押すと、バッチファイルの実行が再開されます。

また、同時に注釈文を入れておくと、止めると同時にStrike a key when readyの前にその注釈文を表示します。

REM (remark)

（レム）

バッチファイル実行中に、注釈文を表示する。

形　式

REM␣[注釈文]

働きと使いかた

バッチファイルの実行がこの命令まで来ると、後につけられている注釈文を表示します。
注釈文の中では、スペース、タブ、カンマ以外の区切り記号は使わないでください。

実行例

```
REM differential equation
```

バッチファイルの実行がここまで来ると、differential equation という文字列を表示します。

REN (rename)

リネーム

ファイル名を変える。

形　式

REN␣[ドライブ名]ファイル名.タイプ名␣新しいファイル名.タイプ名

ドライブ名	条 A:、B:、C:、D:、E:、F:、G:、H:
	省 現在使用中のディスクドライブ
ファイル名	条 8字以内の文字列
タイプ名	条 3字以内の文字列

働きと使いかた

ファイル名やタイプ名を変えます。ファイル名やタイプ名にはワイルドカードが使えます。新しいファイル名やタイプ名にワイルドカードを使うと、その部分は変更されません。

実行例

```
REN *.LST *.PRN
```

使っているドライブの.LSTというタイプ名のファイルを全部.PRNというタイプ名にします。

```
REN B:SUN ?O?
```

ドライブBのSUNというファイルをすべて、SONというファイル名にします。

TIME (time)

タイム

時刻を表示し、設定する。

形　式

TIME␣[時 [：分 [：秒]]]

時：分：秒　条 時：0≦時≦23の整数

分：0≦分≦59の整数

秒：0≦秒≦59の整数

働きと使いかた

TIME [RETURN] を実行すると次のメッセージが表示されます。

Current time is　時：分：秒．1/100秒

Enter new time：

メッセージに表示された時刻を変更しなくてよいときは、[RETURN] キーを押します。

時、分、秒の間は、コロン(:)で区切ってください。入力がまちがっていると

```
Invalid time
Enter new time:_
```

が表示されます。

- 分や秒を省略すると、00が自動的に設定されます。
- 時計の機能のないコンピューターの場合、このコマンドは実行しても意味がありません。

実行例

```
TIME 3:06:22
```

TYPE（type）

指定されたファイルの内容を画面に表示する。

形　式

TYPE␣[ドライブ名]ファイル名．タイプ名

ドライブ名　条 A：、B：、C：、D：、E：、F：、G：、H：
　　　　　　省 現在使用中のディスクドライブ
ファイル名　条 ８字以内の文字列
タイプ名　　条 ３字以内の文字列

働きと使いかた

アスキー形式のファイルの内容を見るためのコマンドです。ファイル名を見つけるには、DIRコマンドを使います。

●機械語のファイルの内容は、このコマンドでは見ることができません。

実行例

```
TYPE A:ROME.C
```

AドライブのROME.Cというファイルの内容が表示されます。

3-786-731-01 (1)

MSX-BASIC Version 3.0

文法書

ソニー株式会社　〒141 東京都品川区品川6-7-35

お問い合わせはお客様相談センターへ
●東京(03)448-3311 ●大阪(06)251-5111 ●名古屋(052)232-2611

Printed in Japan